Technical Report

UCAM-CL-TR-19
ISSN 1476-2986

Number 19

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Analysis and inference for English

Arthur William Sebright Cater

September 1981

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

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

© 1981 Arthur William Sebright Cater

This technical report is based on a dissertation submitted September 1981 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Queens' College.

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

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

ISSN 1476-2986

Table of Contents

Chapter 1: Introduction

1.1) The form and focus of the project.

AD-HAC is a computer program which understands stories. The name AD-HAC is intended to reflect two of its characteristics: firstly, many of its processes are unashamedly 'ad hoc', reflecting my belief that finding a workable solution to a problem is a good first step towards finding an ideal solution to that problem; secondly, its development has proceeded by repeatedly adding 'hacks'. This approach has resulted in a surprisingly robust program, whose degree of understanding enables it to answer simple questions about very short stories, and also to identify pronoun referents in those stories. It accepts the text of the stories, and of the questions, in ordinary English, and answers the questions in English.

The two tasks of question answering and pronoun resolution both require some ability to draw inferences from a text. More generally, for texts accepted as coherent, inferences are required to establish the underlying connections which make the text coherent; only by finding these connections can comprehension be achieved. Indeed, inference is often necessary even for the satisfactory processing of single sentences. I take it as self-evident that inferential processes must act upon some representation of the content or meaning of a text or utterance, rather than upon, for example, a purely syntactic or surface lexical characterisation; and this implies that the nature of the representation is important. The representational language developed for the project has been based upon Schank's Conceptual Dependency, which assumes that a text meaning representation is conceptual rather than linguistic.

AD-HAC contains three largely independent programs: a sentence analyser, a sentence generator, and an inference mechanism. The project has focussed on the problems of analysis and inference, with little emphasis on generation. The sentence analyser is a novel blend of several existing ideas. It exploits the notion of preference, using requests to build up a conceptual analysis, and it uses an augmented transition network to locate simple syntactic constituents. The analyser is essentially nondeterministic, but this nondeterminism is restricted because the analyser operates with a small bounded memory for partial analyses.

The knowledge used by the inference mechanism is represented by a set of inference networks. Many of these networks are associated with the conceptual primitives of the underlying representation language, and are used to draw low-level inferences; some of the networks encode higher-level, i.e. script-like, knowledge, but these script networks play a subsidiary role, being accessed from the low-level networks when appropriate. The entire system consequently has the ability to exploit script-like knowledge, but does not rely upon scripts for its performance.

The program is intended to perform the tasks of pronoun resolution and question answering, rather than to model the way humans perform them. Psychological considerations have not played a very important role in the construction of the program, though there are many aspects of the program which seem psychologically plausible. It must however be emphasised that the program is not intended as a cognitive model.

1.2) Whys and wherefores

There are clearly many areas where the introduction of computers would ease, or even revolutionise, the work that must be done in everyday working life, and in many of these areas the only obstacle is the awkwardness of the man-machine interface. For such applications as these, the ability of the computer to converse in a natural language (NL), such as English, would be a great asset. Thus, a strong motivation for attempting to create an automatic natural language processor (NLP) rests on the usefulness of such a processor.

There are many language-using tasks to which the ideal NLP would be applicable: translating between languages, abstracting scientific papers, composing letters, etcetera. As a first step towards this ideal, we might attempt to build a NLP which was as competent as a typical person in its use of language: in particular, we might try to model how people use language.

At present, the construction of such a model seems inconceivable. Technical problems alone - the problems of computer speed and memory size - are enough to rule out the realisation of this dream for many years. More importantly, there remain a host of theoretical problems of all kinds and on all fronts, many of which cannot even be effectively stated at this time. A taste of the variety of these problems may be gained from the brief list below.

How should auditory patterns be stored in machine memory?
What role does syntax play in the understanding process?
Why do people produce grammatically incorrect sentences?
Can the various senses of a single word be enumerated?
How should knowledge of the world be represented?
Is a language's vocabulary finite?
How can metaphors be understood?
What is consciousness?

Theoretical problems such as these are interesting in their own right: and even though they cannot yet be studied practically in the context of an overall computer model of language use, it is possible to examine some of these topics in relative isolation, by constructing partial models. In doing so, it is necessary to make assumptions about the more general model, where these assumptions may themselves provide insights into other problem areas.

One particularly common partitioning of language use into subproblems is the division between speech processing and written text processing. Computer models of text comprehension may ignore many aspects of spoken language, such as the representation and use of phonetic information, or the interpretation of certain cues - such as pauses, "um, er" - which do not occur in texts; though a model of text comprehension will not be a satisfactory cognitive model if it relies on the exclusion of such aspects. Similarly, a model of speech processing need not consider various text-specific phenomena, such as the extra ambiguity resulting from the absence of cues such as stress and intonation; again, to be adequate as a cognitive model, a speech processor must not rely on the exclusion of these phenomena.

Most work in the field of NLP has been oriented towards text, rather than speech, comprehension. There are several reasons for this. Firstly, computers can easily handle textual material in a form which makes the recognition of characters and words quite trivial, bypassing the real problem of simultaneous deployment of many different sorts of knowledge; secondly, speech input and output requires additional hardware; thirdly, the problems of phonemic recognition appeared quite intractable until relatively recently; and finally, the results of such phonemic recognition would, in the most obvious and straightforward case, be the computer's own representation of the printed word: thus the problems of text comprehension would be augmented with problems of input.

AD-HAC is designed to understand stories presented in the most straightforward form, as textual material presented via a computer terminal. This mode of input permits the problems of speech- or character-recognition to be sidestepped, so that attention may be focussed on the central problems of comprehension; the restriction to stories sharpens up these interesting fundamental problems by permitting their study in a simplified context. Specific reasons for such a restriction include:

i) The provision of enormous quantities of specialised knowledge can be avoided. This would not be the case if, for instance, one were to attempt to abstract scientific papers.

ii) It is unnecessary to cater for exceedingly complex syntactic constructions, but it is advisable to handle a fairly wide range.

iii) Stories naturally possess an event sequence, and provide a relatively self-contained world about which questions may be asked, as in school comprehension tests.

iv) Stories implicitly refer to the types of everyday knowledge possessed by everybody, and thus do not artifically restrict or extend the types of inference needed.

Story understanding is a good task precisely because these simplifications do not alter the nature of the required processes: the right degree of generality is captured. Story understanding requires that global language-using processes, for instance pronoun resolution and question answering, be adequately modelled; but relatively local task-specific processes, e.g. the use of two languages for translation, or the recognition of speech input, do not need to be considered.

1.3) How AD-HAC runs

The implemented program has three principal components, reflecting a similar division of the overall language-processing task. Individual phrases and clauses (conventionally, sentences) forming a text must be analysed; these analyses must be integrated to allow context to influence the understanding of the whole text, and complementarily to complete the understanding of the individual sentence; and sentences must be generated to show understanding, for example as responses to questions.
These components simply fit together as shown below:

```
 __________         ______________         ___________
| Sentence |---->| Inferential  |---->| Sentence  |
| Analyser |     |  Processor   |     | Generator |
 ----------       --------------       -----------
```

In operation, sentences are analysed one-by-one, and initial representations of their meaning are communicated to the inference component. This draws inferences from the initial representations, deriving new information from old, and sometimes calling on the sentence generator to express this information in English. The representation used for communication between the components is uniform, and so the sentence analyser can in practice be linked directly to the generator if this is desired.

The inferencer's representation of a text - its "memory" - takes the form of a number of simple propositions connected together by links of various kinds. Typically, many of these propositions will be very similar, and the inference mechanism tries to compress the network by merging nearly-identical propositions. As a side effect of this compression, the inferencer determines the referents of pronouns and the answers to questions.

The entire program is written in Standard Lisp (sic), and runs on both the IBM370 in Cambridge and the DEC20/50 in University College, Dublin. The version which runs in Cambridge operates in batch mode, since only extremely small programs may be run interactively there: AD-HAC is not a tiny program, but occupies some 850Kbytes on the IBM machine, or some 200Kwords on the DEC. The analysis phase takes up to 5 seconds on the DEC machine for very complex sentences, though the typical time for an individual sentence is less than one second. The inference component takes a lot of time, and gobbles a correspondingly large amount of space: the processing of the story shown in the next section takes about 30 seconds on the UCD DEC machine.

1.4) An example run of AD-HAC

To give some indication of the system's capacity, an example run is shown below. The remainder of the thesis shows how this behaviour is achieved. The system prompts for more input with the phrase "Pray continue:", and marks the beginning and end of the analysis with a "[" and "]" respectively. The rest of the material here should be self-explanatory.

```
*(MAIN I)

Pray continue:
BILL AND JILL WENT TO THE ZOO.
[]

Pray continue:
THEY GAVE THE MONKEYS SOME PEANUTS, WHICH THEY ATE.
[]

Pronoun resolution : IT WAS JILL AND BILL WHO GAVE SOME PEANUTS TO SOME
MONKEYS.

Pronoun resolution : IT WAS THE MONKEYS WHICH ATE SOME PEANUTS.

Pray continue:
THEY WENT TO THE RESTAURANT AND DRANK SOME TEA.
[]

Pronoun resolution : IT WAS JILL AND BILL WHO MOVED TO A RESTAURANT.

Pray continue:
JILL TOOK BILL'S MONEY FROM HIM AND GAVE IT TO THE TRAMP WHO WAS TALKING TO
THEM.
[]

Pronoun resolved : IT WAS BILL FROM WHOM JILL TOOK SOME MONEY.

Pronoun resolved : IT WAS THE MONEY WHICH JILL GAVE TO A TRAMP.

Pronoun resolved : IT WAS JILL AND BILL TO WHOM A TRAMP SAID SOMETHING.

Pray continue:
WHAT DID SHE GIVE HIM?
[]

Pronoun resolved : IT WAS JILL WHO GAVE SOMETHING TO A MALE.

Answering question (1): JILL GAVE SOME MONEY TO A TRAMP.

Pray continue:
(Returning to Lisp...
```

1.5) The organisation of the thesis

The remainder of the thesis gives first the method of representing meaning, which forms the language for communication between the components of the overall program. Later chapters describe in detail the components themselves.

Chapter 2 is devoted to a description of the representation language as it now stands, which is a development of Schank's Conceptual Dependency. In particular, it describes how conceptual primitives are combined to represent a message, which may loosely be defined as the meaning a speaker wishes to convey to his hearer. It also describes how simple messages may be combined to form larger messages, with special emphasis on the uniformity of representation of the meaning underlying various linguistic constructions: this is one area where Schank's original CD was rather weak. The chapter also describes the representation of time, which turns out to be very important to the inference mechanism.

Chapter 3 then describes the sentence generator, which is based on Goldman's BABEL. This was the first component of AD-HAC to be built, and it is also the simplest. The generator is described here for two reasons: firstly, so that the order of development may be reflected in the order of exposition; but secondly, since the generator is relatively simple and unoriginal, its description and illustration may further assist the reader in understanding the representation language.

Chapters 4 and 5 are concerned with the sentence analyser: Chapter 4 gives some background, describing earlier work on analysis, concentrating on earlier systems which were influential in the design of AD-HAC's analyser, namely those of Riesbeck, Wilks and Woods. This chapter concludes by outlining five principles which should be noted in designing a sentence analyser. Chapter 5 describes AD-HAC's own analyser, showing how these five principles have guided its design.

Chapter 6 gives an annotated run of the program, concentrating on the rôle of the sentence analyser, but also giving a preview of the workings of the inferencer.

Chapters 7 and 8 discuss inference: first, Chapter 7 gives background, giving particular attention to the systems of Charniak and Rieger because these had an especially strong influence on the development of AD-HAC; Chapter 8 describes AD-HAC's inference mechanism, showing how inference networks are constituted, how they function, and how background processes maintain extensive cross-reference information which is critical to AD-HAC's pronoun resolution and question-answering behaviour.

Chapter 9 gives another annotated run of the program, focussing this time on the inference mechanism.

Chapter 10 describes some relevant parallel work which has taken place during AD-HAC's development: the work discussed there did not influence AD-HAC's design, but is of interest because it addresses many of the same issues.

The concluding chapter, Chapter 11, examines the successes and failures of AD-HAC, drawing the usual lessons for the future.

Chapter 2: Representation

2.1) Requirements for a representational system

When a speaker makes an utterance, his primary aim is usually to communicate some message. His hearers have the task of extracting that message from his utterance; they use syntactic knowledge to help them do this, but their main interest is in the content rather than the form of the utterance.

For any NL processing system, the internal language of that system must represent messages, rather than the superficial textual form of their corresponding utterances. Ideally, the internal language should also facilitate all operations which must be performed by the system; namely analysis of input natural language texts into internal language, generation of natural language from this language, and the various representation-dependent operations necessary for inference, such as assembling, dissecting and matching the internal forms of messages.

The most critical of these operations is matching internal forms of messages: for upon this capability will rest the ability of an inference mechanism to recognise, and hence merge, similar messages. This consideration immediately implies that the internal language should exhibit two features: first, each message should have a small number of possible internal forms (ideally only one) ; and second, each internal form should have only one interpretation.

This in turn suggests that the internal language should be based upon symbols which represent elements of meaning common to a group of words. For instance, there is some similarity of meaning between the words 'run' and 'walk', and a symbol can be designated to capture this similarity. Such a symbol is called a semantic primitive. The difference between 'run' and 'walk' may be captured by the concurrent use of other symbols. Yet other symbols may be used to denote the common meaning of deep relationships: for instance a cause/effect relationship may usefully be denoted by some symbol. It is irrelevant whether these symbols are themselves words in some natural language: for they will not be manipulated in the same fashion as the words of the natural language with which the system is dealing.

Other considerations may influence the design of a representational system, though they should not be permitted to compromise the principal goal of satisfactorily representing a wide variety of texts. The processes of natural language analysis and generation are facilitated if the internal language captures some of the regularities of natural languages. For instance, in the course of the work described here, the representation of modifiers has been found very important. For this and other purposes, an intuitive classification of messages into EVENT, STATE and CAUSE relations appears to work very well. The representation system adopted is based on Schank's Conceptual Dependency (CD). Though there has been some development, the system used will continue to be referred to as CD in the sections which follow. References to CD should be understood as referring to the representation system used in AD-HAC, except when, as in the next section, the context clearly indicates that Schank's original system is meant. What has hitherto been called "internal form" will now be denoted by the term CDform.

2.2) Schank's Conceptual Dependency

Conceptual Dependency [Schank, 1972], [Schank, 1973], is a representational language based on the use of conceptual primitives and conceptual cases. Each primitive is associated with a particular set of cases: these cases may be thought of as the names of slots which must be filled by other conceptual entities.

The term conceptualisation - or sometimes conceptual diagram - is used by Schank to mean a pattern in which one of these primitives is combined with fillers for each of its cases: these fillers may themselves be conceptualisations, but are often picture producers, which are conceptual entities which usually correspond to concrete nouns in English. The picture producers have an internal structure, which permits them to encode the information provided in English by adjectives, possessives and relative clauses.

2.2.1) In praise of the principles of Conceptual Dependency

CD maintains a distinction between the notions of EVENT, STATE and CAUSE. The conceptual primitives fall into two classes, the primitive ACTs and the names of primitive STATEs, while causal relationships are encoded by the use of subordinating roles.

The number of ACTs proposed by Schank is very small: the precise number has fluctuated between 11 and 16, as the need for further acts was noticed, or as new generalisations were found. The significance of these acts is twofold. Firstly, each ACT needs a specific set of accompanying cases in order for a valid, meaningful, conceptualisation to be formed; this means that an inference mechanism can easily spot where further information is needed, and can attempt to fill the corresponding slots with appropriate information. Secondly, each ACT naturally indicates the appropriateness of a specific set of inferences; these inferences, it has been argued, are what make the ACT itself meaningful.

The number of states used in CD is relatively open-ended. Nevertheless, each state leads to a set of inferences in the same way as the ACTs. The distinction between states and events has some external justification, because it has been shown that a simple causal syntax exists when inferences are made from conceptualisations: the "results" of an event occurring may be represented either as new states arising, or as changes of some existing state; the "results" of a state being true may be represented as the enablement, or prevention, of some event.

Perhaps the most important merit of CD is that it encourages canonical representation, because the number of primitives is very small. The main criticism of the next section is that Schank's CD did not take full advantage of this: the subsumption of many forms of linguistic modification into "picture producers" seems to me to be mistaken. However, the choice of primitives in CD very nearly permits the one-to-one correspondence between messages and CDforms which is so desirable a property for the purposes of matching. The CD representation of a message as "a reciprocal ACTOR-ACT dependency", with subsidiary information according to the ACT primitive, permits the analyser, the generator, and the various inferential processes to exploit this implicit information when decisions must be made.

2.2.2) Criticisms of Schank's implementation of Conceptual Dependency

Several criticisms of CD can fairly be made. Principal amongst these is the multiplicity of mechanisms for representing basically similar information in different contexts. Contexts in which the same information is present, in grossly different formulations, are shown as Figure 2.1. The cause of this difference is a lack of structural devices within the representational language.

```
---------------------- Figure 2.1 ----------------------

 "The hat is red"              "The red hat ..."

 (CON (ACTOR (£PHYSOBJ         (CON (ACTOR (£PHYSOBJ
              TYPE (*HAT*)                  TYPE (*HAT*)
              ... )                         COLOUR (*RED*)
       IS (COLOUR                           ... )
              VAL (*RED*))           ... )
       TOK CON1))

-------- nonuniform representation in Schank's CD --------
```

A related criticism concerns the representation of relative clauses; whilst the presence of "(REL <conceptualisation>)" is strictly sufficient for the purposes of an inference mechanism, it seems quite clumsy. A more elegant approach is to identify the place where the relativised nominal occurs, thus eliminating the fishing about within structures that must be done if the only information is encoded by REL. This becomes especially important for deeply embedded relative clauses; eg "The banana I asked you to give the man"

A third criticism of CD, as developed by Schank et al, concerns the representation of time information. A conceptualisation has two parts: the "core" message, and an unstructured list of relationships between the time tokens occuring in that message. This seems sufficient, but clumsy. Where conjunctions such as 'WHEN' or 'BEFORE' are used to relate two clauses, the conceptual representation of the corresponding sentence would be closely related to that for 'AND'; I have doubts about even the sufficiency of this scheme for handling such sentences as

"Did John eat breakfast before going to work?"

With the introduction of some structuring tools, the internal language should be capable of representing all relevant information in one place.

Lastly, the use of (REF A) or (REF THE) to pass information between an analyser and a generator seems misplaced. Recent literature on the subject of definite reference indicates that such reference signifies that the hearer should be able either to identify the referent, or to recognise that the referent need not be identified. In either case, the provision of (REF THE) or (REF A) is merely skating around the problem faced by a natural language generator.

2.2.3) Summary of the original implementation

CD is a well motivated representation language: the combination of primitive terms with specific sets of roles, corresponding to conceptual case frames, and the ability to nest conceptualisations within one another, make it easy to use as a formalism for inferential processes. Indeed, it continues to be used by the Yale AI group, and forms the basis of several experimental systems - SAM, PAM, QUALM, TALE-SPIN, and others.

The main problem with the original implementation was that the same information could be represented in different ways depending on the linguistic context in which it was presented: there was one way of representing conceptual information as conceptualisations, and another as picture producers. Conceptualisations were usually used to represent the information conveyed by a clause, while picture producers were used to represent the information in noun phrases.

The problems raised above are not fatal defects, but are flaws of detail. The challenge of implementing a new natural language processing system, without any preexisting software on which to build, brings with it the opportunity to tackle these problem areas in a new way.

2.3) A new implementation of Conceptual Dependency.

The discussion above has been concerned with Schank's Conceptual Dependency, and specifically with its implementation in such projects as MARGIE, SAM, PAM and TALE-SPIN. My criticisms of that implementation are concerned with some detailed and relatively obscure aspects: they have provided the impetus for me to implement a related internal representation language, based on the same fundamental precepts but attempting to avoid these criticisms.

2.3.1) The new format of CDforms

CDforms are labelled as EVENT, STATE, CAUSE or CONJUNCT, and consist of a number of filled slots, which are also labelled. The top-level labelling is actually superfluous, because each EVENT has an ACT and ACTOR slot, each STATE has a STATENAME slot, each CAUSE has an ANTECEDENT and a RESULT slot, and each CONJUNCT has a FIRST and SECOND slot; the presence of these top-level labels does however facilitate certain detailed aspects of the necessary programming.

2.3.1.1) EVENT representation

Schank's CD concentrated largely upon the set of ACT primitives, and this set has been retained. The 14 ACTs used in AD-HAC are:

ACT?	ATRANS	ATTEND	DO	EXPEL
GRASP	INGEST	MBUILD	MFEEL	MOVE
MTRANS	PROPEL	PTRANS	SPEAK	

A good discussion of the significance of these primitives (excepting ACT?) is found in [Schank, 1975]. "ACT?" is not one of Schank's original acts; it has been introduced simply to have some representation for such sentences as "WHAT DID FRED DO?", and is the mark of an outstanding problem in the new CD.

The set of acts is of especial importance in CD, for several reasons. The ACT of an EVENT is analogous to a verb in most sentences of a natural language: it provides the strongest influence on the production of inferences (as will be seen in Chapters 7 and 8), and hence contributes most to the meaning; it selects a set of subsidiary roles for the EVENT, and constrains their possible fillers.

Some annotated examples of simple EVENTs follow.

```
                    "JOHN WENT TO THE PARK"
((EVENT (ACTOR JOHN)                       ; An event performed by JOHN
        (ACT PTRANS)                       ; doing some movement
        (OBJECT JOHN)                      ; of JOHN
        (FROM DUMMY-PLACE1)                ; from somewhere
        (TO PARK1)                         ; to a particular park
        (TIME (NAMED TIMEPOINT1)           ; at some time
              (COMPARISON                  ;    that time being
                 (BEFORE *NOW*)))))        ;    in the past
```

```
                    "HE TALKED TO MARY"
((EVENT (ACTOR DUMMY-MALE1)                ; An event performed by some male
        (ACT MTRANS)                       ; communicating
        (MOBJECT CONCEPTS1)                ; some unspecified information
        (FROMCP DUMMY-MALE1)               ; from his "conscious part"
        (TOCP MARY)                        ; to MARYs "conscious part"
        (TIME (NAMED TIMEPOINT2)           ; at some time
              (COMPARISON                  ;    that time being
                 (BEFORE *NOW*)))))        ;    in the past
```

Each EVENT has, besides an ACT, an ACTOR and a TIME. Other roles may appear as appropriate for the chosen ACT - thus, in the examples above, the act PTRANS introduces the additional roles OBJECT, FROM and TO; and the act MTRANS introduces MOBJECT, FROMCP and TOCP.

The roles TRUTH and ABILITY may also appear within an EVENT, but they are mutually exclusive. The possible modifications of an EVENT are:

```
(TRUTH TRUE)       - the event occurred
(TRUTH FALSE)      - the event did not occur
(TRUTH TRUTH?)     - the occurrence is being questioned
(ABILITY CAN)      - the event could occur
(ABILITY CANNOT)   - the event could not occur
(ABILITY ABILITY?)- the possibility of the event is being questioned
```

If, as above, no such modification is found, (TRUTH TRUE) is assumed. As with the top-level labelling of structures as EVENT, STATE etc., the provision of these distinct labels is superfluous, since the appropriate label could be deduced from the content: and again, the redundancy facilitates certain aspects of the implementation.

As an example of the use of the ABILITY modifier, the representation shown overleaf is used for the sentence "JOHN CAN GO TO THE PARK".

```
((EVENT (ACTOR JOHN)                          ; Previously, if no other
        (ACT PTRANS)                          ; information was available,
        (OBJECT JOHN)                         ; TRUTH = TRUE was adopted as a
        (FROM DUMMY-PLACE2)                   ; shorthand. Here, other info
        (TO PARK1)                            ; is available: viz.
        (ABILITY CAN)                         ;    The action CAN be performed.
        (TIME (NAMED TIMEPOINT4)              ;
              (COMPARISON (AFTER *NOW*)))))
```

2.3.1.2) STATE representation

CD distinguishes EVENTs and STATEs: besides an inventory of ACT primitives, there is a selection of STATENAME primitives, though this is less clearly defined. There is also a subcategorisation of states, into scale-states and non-scale-states. The scale states are used to represent the sort of static information which is commonly conveyed by adjectives in natural languages. AD-HAC currently uses the scales listed below:

AGE	eg for YOUNG,MIDDLE-AGED,OLD,ANCIENT
ANGER	for TRANQUIL,CALM,ANNOYED,FURIOUS,LIVID
ANXIETY	for CAREFREE,ANXIOUS,FEARFUL
AWARE	for SLEEPING,RESTLESS,DROWSY,AWAKE
BENEFIT	for no adjectives: see below
FEAR	for CAREFREE,CALM,NERVOUS,FRIGHTENED,SCARED
GUILT	for OBDURATE,IMPENITENT,REMORSEFUL,GUILT-STRICKEN
HEALTH	for DEAD,CRIPPLED,AILING,SICK,WELL,HEALTHY,FIT
HUNGER	for BLOATED,SATED,SATISFIED,PECKISH,HUNGRY,RAVENOUS,STARVING
IQVAL	for IDIOTIC,STUPID,DUMB,SILLY,SMART,CLEVER,BRAINY
JOY	for DEPRESSED,SAD,UNEMOTIONAL,HAPPY,ECSTATIC
LENGTH	for SHORT,AVERAGE-LENGTH,LONG
PSTATE	for RUINED,BROKEN,DAMAGED,INTEGRAL
QUALITY	for ABYSMAL,LOUSY,BAD,POOR,REASONABLE,GOOD,EXCELLENT
SIZE	for MINUTE,TINY,LITTLE,SMALL,AVERAGE,BIG,HUGE,ENORMOUS
SOBER	for PISSED,DRUNK,TIPSY,SOBER
SOCGRACE	for SCATHING,RUDE,IMPOLITE,OFFHAND,POLITE,OBSEQUIOUS
THIRST	for SLAKED,DRY,THIRSTY
WEIGHT	for FEATHER-LIGHT,LIGHT,UNHEAVY,HEAVY,PONDEROUS

A scale-state always uses the roles STATENAME, THING, VAL and TIME. TRUTH may also be specified, as with EVENTs, but ABILITY may not. In a scale-state representation, VAL indicates a place on a notional scale from -10 to +10; THING identifies the object - or person - which is characterised by being at this position on the scale. The example adjectives given above for each scale name correspond to progressively higher numbers in the VAL roles; the scale BENEFIT, however, has no adjectival correlates in English: it is used in a fuzzy fashion which is explained later in section 2.3.4. As a simple example of the use of scale states:

```
                    "THE TABLE IS BIG"
((STATE (THING TABLE1)                        ; A particular table is
        (STATENAME SIZE)                      ; characterised by having size
        (VAL 4)                               ; somewhat bigger than average
        (TIME (NAMED *NOW*))))                ; at this particular time.
                                              ; (Inference will establish that
                                              ; no change is likely over time)
```

In scale-state representations, VAL may alternatively encode a change of state. This is achieved by having, eg, (VAL (HIGHERBY 2)) rather than, eg, (VAL 2). A similar mechanism existed in Schank's CD. There is a further option in the new CD: the VAL of a scale-state may be declared to be somewhere in a given range of values, by use of the special forms "(VAL (HIGHERTHAN n))" and "(VAL (LOWERTHAN n))". In particular, this may be used to represent the meaning of comparative adjectives; precisely how this can be done is shown in section 2.3.3.

There are also several non-scale-states. Several new states have been added to the set originally proposed by Schank; the full set now used is:

COLOUR	EXIST	FAMILY	GOAL	IDENTITY
ISA	LOC	MLOC	OWN	PART
POSS	QUANTIFY	QUANTITY		

These, like the ACTs, introduce a specific set of subsidiary roles. Many of them also place strict requirements upon the fillers of these roles: for instance, QUANTIFY uses the roles THING and VAL, but VAL must be one of a special set of tokens: eg. *EVERY* *MANY* *NO*. Some annotated examples of state-based CDforms are shown below:

"THE BLOCK IS ON THE TABLE"

```
((STATE (THING BLOCK1)                   ; A particular block has a
        (STATENAME LOC)                  ; location relative to
        (VAL TABLE2)                     ; a particular table
        (SPATREL *ON*)                   ; the spatial relation being ON
        (TIME (NAMED *NOW*))))           ; at this particular time.
```

"THE LAMB BELONGS TO MARY"

```
((STATE (STATENAME OWN)                  ; An ownership relation holds,
        (THING LAMB1)                    ; where the lamb is owned
        (VAL MARY)                       ; and MARY is the owner
        (TIME (NAMED *NOW*))))           ; at this time.
```

"FRED KNEW THAT MARY HAD GONE TO THE SHOPS"

```
((STATE                                  ;
  (STATENAME MLOC)                       ; Mentally located
  (INCP FRED)                            ; in FRED'S "conscious part"
  (TIME (NAMED TIMEPOINT1)               ; at some time
        (COMPARISON (BEFORE *NOW*))      ;    that time being in the past
  (MOBJECT                               ; was the information that
     (EVENT (ACTOR MARY)                 ; MARY performed the action
            (ACT PTRANS)                 ; of moving
            (OBJECT MARY)                ; MARY herself
            (FROM DUMMY-PLACE1)          ; from somewhere
            (TO SHOPS1)                  ; to some set of shops
            (TIME (NAMED TIMEPOINT2)     ; at some time
                  (COMPARISON            ;    that time being
                     (BEFORE *NOW*))     ;    in the past
                  (COMPARISON            ;    and also before the time of
                     (BEFORE             ;    the information being thus
                        TIMEPOINT1)))))) ;    located. (Not implying
                                         ;    before all such times)
```

2.3.1.3) CAUSE representation

In the original implementation of CD, causal relationships were indicated by the use of subordinating roles: thus an EVENT conceptualisation might have, besides the roles for ACTOR, OBJECT etc., an extra role RES, indicating a result of that event. There were in fact several such roles corresponding to different types of causality.

In this new implementation, CAUSE may appear as a top-level label on a conceptualisation: it has two roles, ANTECEDENT and RESULT, both of which subordinate other conceptualisations. For example,

"FRED MADE MARY ANGRY"

```
((CAUSE
    (ANTECEDENT
       (EVENT (ACTOR FRED)
              (ACT DO)
              (TIME (NAMED TIMEPOINT1)
                    (COMPARISON (BEFORE *NOW*)))))
    (RESULT
       (STATE (STATENAME ANGER)
              (THING MARY)
              (VAL 4)
              (TIME (NAMED TIMEPOINT1))))))
```

Additionally, a CAUSE can have the roles ABILITY or TRUTH, though as in the case of events, these two roles are mutually exclusive. A CAUSE CDform may arise in several ways: when the word "BECAUSE" appears in a sentence is one obvious way; many verbs, such as "MAKE" in the sense illustrated above, but also verbs such as "HIT", "PREVENT", "ADVISE", demand the representation of a causal relationship. One surprising use of causality occurs in sentences such as

"JOHN CAN BE HAPPY"

```
((CAUSE                                    ; A causal relationship
    (ABILITY CAN)                          ; CAN hold between
    (ANTECEDENT CONCEPTS2)                 ; some unspecified cause
    (RESULT                                ; and the result of
       (STATE (THING JOHN)                 ; JOHN being characterised by
              (VAL 4)                      ; a higher-than-average position
              (STATENAME JOY)              ; on the JOY scale.
              (TIME (NAMED TIMEPOINT5)
                    (COMPARISON (AFTER *NOW*)))))))
```

This arises because the role ABILITY applies only to events and causes, but not to states; even though the English sentence superficially demands that the state be given an 'ability' marker, the semantics of the representation language forbid this. It can be seen, therefore, that the conceptual primitive 'CAN' is not the same as the English word "CAN", and the primitive 'CANNOT' is not the same as the word "CANNOT".

One problem with analysing English sentences into this representation concerns the interpretation of negation when applied to causal relationships. For instance, which is the appropriate paraphrase for the sentence "FRED DIDN'T ANNOY MARY BY EATING THE BISCUITS"

- Fred's eating the biscuits was not what annoyed Mary.
- Fred didn't eat the biscuits, but Mary became angry.
- Though Fred ate the biscuits, Mary did not become angry.
- Fred didn't eat the biscuits, and Mary didn't get angry.

These four cases correspond to different ways of placing the negative in the CDform, and the current analyser cannot select between the four options:

- negating the CAUSE
- negating the ANTECEDENT
- negating the RESULT
- negating both ANTECEDENT and RESULT

Consequently, the representation language has an optional role so that the analyser may pass to the inferencer a representation which leaves this ambiguity unresolved; the structure passed for the above sentence would be

```
((CAUSE
    (ANTECEDENT
       (EVENT (ACTOR FRED)
              (ACT INGEST)
              (OBJECT BISCUITS1)
              (TIME (NAMED TIMEPOINT1)
                    (COMPARISON (BEFORE *NOW*)))))
    (RESULT
       (STATE (STATENAME ANGER)
              (THING MARY)
              (VAL 4)
              (TIME (NAMED TIMEPOINT1))))
    (UNKNOWN (TRUTH FALSE))))
```

It is to be hoped that later incarnations of the inferencer will be able to resolve this ambiguity; at present, this cannot be done.

2.3.1.4) The representation of conjunctions

The new implementation of CD has three ways of handling conjunction, because there seem to be three different types of conjunction:
- conjoined clauses, as in "FRED WENT TO THE KITCHEN AND ATE A BANANA"
- conjoined nominals, as in "FRED AND MARY WENT TO THE ZOO"
- conjoined modifiers, as in "FRED ATE A GREEN AND UNRIPE BANANA"

Conjoined clauses are handled by means of CONJUNCT conceptualisations; these take the roles FIRST and SECOND. For example,

"FRED WENT TO THE KITCHEN AND ATE A BANANA"

```
((CONJUNCT
    (FIRST  (EVENT (ACTOR FRED)
                   (ACT PTRANS)
                   (OBJECT FRED)
                   (TO KITCHEN)
                   (TIME (NAMED TIMEPOINT1)
                         (COMPARISON (BEFORE *NOW*)))))
    (SECOND (EVENT (ACTOR FRED)
                   (ACT INGEST)
                   (OBJECT BANANA1)
                   (TIME (NAMED TIMEPOINT2)
                         (COMPARISON (AFTER TIMEPOINT1))
                         (COMPARISON (BEFORE *NOW*)))))))
```

Conjoined nominals are handled by the use of a GROUP as the filler of a role. A group has any number of elements, and these are labelled 1,2, etc. For example,

```
                        "FRED AND MARY WENT TO THE ZOO"
((EVENT (ACTOR (GROUP (1 FRED)
                      (2 MARY)))
        (ACT PTRANS)
        (OBJECT (GROUP (1 FRED)
                       (2 MARY)))
        (TO ZOO1)
        (TIME (NAMED TIMEPOINT1)
              (COMPARISON (BEFORE *NOW*)))))
```

Conjoined modifiers are handled by nesting one conceptualisation inside another, in such a way that the particular object being modified is pointed out within the embedded conceptualisation. This is achieved by use of the FOCUS device, which is explained in section 2.3.3 below.

2.3.2) The conceptual roles

CDforms have the structures illustrated in the preceding subsections. They comprise a set of role names and associated fillers, and this whole assemblage is labelled with its type - EVENT, STATE, CAUSE or CONJUNCT. These types demand that certain roles should appear: thus, all EVENTs have an ACT and an ACTOR; all STATEs have a STATENAME; all CAUSEs have an ANTECEDENT and a RESULT; all CONJUNCTs have a FIRST and a SECOND.

Beyond these mandatory roles, individual ACTs and individual STATENAMEs require further roles. The situation is further complicated by the fact that some pairs (or triples) of roles are mutually exclusive, and some optional roles also exist. This section tries to give a comprehensive picture of the status of the conceptual roles. The sets of roles which may appear with given ACTs and STATENAMES is given in a more compact format in Appendix A.

TRUTH and ABILITY

The role TRUTH may appear in any conceptualisation, and may be filled with the values TRUE, FALSE or TRUTH?. An alternative, in EVENTs and CAUSEs but not in STATEs, is the role ABILITY, which may take the tokens CAN, CANNOT or ABILITY?. If neither of these roles actually appears, TRUTH is assumed to have the value TRUE.

In the special case of CAUSE CDforms, the role UNKNOWN may appear, and will contain as its filler either the expression (TRUTH FALSE) or (ABILITY CANNOT).

TIME

All events, and all states with the exception of IDENTITY, must have a TIME role. The filler of this role must be either a TIMEPOINT or a TIMESPAN; these fillers have a detailed internal form, and are discussed further in section 2.3.5.

ACTOR

All EVENT CDforms have an ACTOR role: the nature of the filler of this role depends on the particular ACT, but is in all cases a conceptual token rather than an embedded conceptualisation.

INST

EVENT CDforms may optionally have an INST role. If it occurs, it must be filled by another EVENT having the same ACTOR and the same TIME.

OBJECT

Almost all EVENTs have an OBJECT role, exceptions being MFEEL, MTRANS, MBUILD and SPEAK; the primitive acts DO and ACT? make take, but do not need, this role. The object role, where it appears, is a conceptual token, though in the case of the act ATTEND it must be one of a small number of sensory organs, eg. *EYE*, *EAR*.

THING and VAL; also RELATION and SPATREL

Each of the 19 scale states listed earlier needs the roles THING and VAL; for these states, which correspond to many simple adjectives, The THING role is filled by some conceptual token, eg. FRED or TABLE1, while the VAL role is filled with one of the forms:

(VAL <number>)

(VAL (HIGHERBY <number>)) or (VAL (LOWERBY <number>))

(VAL (HIGHERTHAN <number>)) or (VAL (LOWERTHAN <number>))

The roles THING and VAL are also used by most of the other states (the exceptions being EXIST, GOAL and MLOC): in each case, THING will still refer to a conceptual token, but the filler of the VAL role will be of a type determined by the particular STATENAME at hand. The roles, and the types of their fillers, are associated with particular states as follows.

COLOUR needs a THING and a VAL, where the VAL will be the token corresponding to the name of a colour - eg RED.

EXIST needs only a THING, which must be a physical object.

FAMILY needs a THING, a VAL (which must be an animate entity of the same type as the THING), and the additional role RELATION, which must be the name of a familial relationship - eg SISTER.

IDENTITY needs a THING and a VAL; it does not matter which is which since identity is a transitive relationship.

ISA needs a THING and a VAL, both of which are conceptual tokens. This state is not actually used in the system, but is provided as a hook on which future extensions of the system may be hung; specifically, metaphorical statements such as "FRED BLOGGS IS A FOX".

LOC needs a THING and a VAL, and has an optional role SPATREL. Where this role is present, it holds one of the tokens which designate spatial relationships, such as *ON*, *BELOW*. Where this role is absent, the entire LOC state merely means that THING is near VAL: nearness, like identity, it a transitive relationship, and so it does not matter which object or place fills the THING role, and which the VAL. Both THING and VAL are constrained to be either physical objects or places; where the role SPATREL is present, its filler will further constrain these in an obvious fashion.

OWN, PART and POSS each take both THING and VAL, and provide distinct representations for the various forms of relationship expressed by possessives in English. The VAL in OWN and POSS is constrained to be a beast, whereas in PART it may be any physical object. The THING must be a physical object in all three cases, and represents the thing possessed.

QUANTIFY needs a THING and a VAL; the THING role must be filled by a (plural) token representing physical objects, while the VAL must be one of the tokens representing quantities.

QUANTITY also needs a THING and a VAL, but in this case the THING may relate to any physical object, while the VAL represents some measure of quantity, eg. *BUCKETFUL*.

MOBJECT and GOALSTATE

The role MOBJECT is used to subordinate an entire conceptualisation, specifically one which corresponds to some information. Thus, in an MTRANS event, it shows the information communicated; in an MBUILD event, it shows the information constructed or deduced; in an MLOC state, it shows the information held in some appropriate repository.

The role GOALSTATE occurs only in a GOAL - one of the primitive states. Like MOBJECT, it holds an embedded conceptualisation, but this is constrained to be a state. The statename GOAL is actually redundant, since the same information could be represented as an MLOC state, where the MOBJECT is a CAUSE relationship between the desired state and resulting BENEFIT for the person (or beast). The use of GOAL, however, introduces a welcome abbreviation into this form of representation.

INCP, INLTM, IN, and WANTER

In the original implementation of CD, higher animates were endowed with a CP and an LTM: these were "Conscious Processor" and "Long Term Memory" respectively, and they served as the repositories, sources and destinations of information. The use of picture producers for concrete nouns was applied to the representation of these hypothesised entities also; thus, if "John told Mary ...", the CD representation would include the elements

(FROM (*CP* PART JOHN)) and (TO (*CP* PART MARY))

In this new implementation, these entities are subsumed into special roles, providing a further abbreviation of the CDforms. The state MLOC uses one of the roles INCP, INLTM and IN, to reflect the information store in which the MOBJECT is held; a given CDform may use only one of these roles: they are mutually exclusive.

The role IN is used when the information is held in some inanimate entity, for example a book or a tape recording.

The role WANTER is used only in a GOAL state, and indicates the (necessarily animate) being which desires some state.

FROMCP, FROMLTM, and FROM; TOCP, TOLTM, and TO

In similar fashion, these roles are used to indicate the sources and destinations of information in an MTRANS event; again, only one of FROMCP, FROMLTM and FROM may be used in a given CDform, and only one of TOCP, TOLTM and TO. Also, FROM and TO are used when a book, for instance, is the relevant filler.

FROM and TO

The acts ATRANS, PTRANS, and PROPEL use the roles FROM and TO, for destination and source respectively. The fact that these roles are not free-standing primitives, but occur only inside whole CDforms, means that the difference in nature between the FROM in an ATRANS and the FROM in a PTRANS does not need to be reflected in having different names for the roles.

Additionally, the acts ATTEND and MFEEL use the role TO. In the case of ATTEND, the TO role indicates the object or place to which a sensory organ is directed; in the case of MFEEL, it indicates the (usually animate) object towards which some emotion is felt.

Other special-purpose roles

There are several roles which are specific to a single act or state. Some of these, WANTER, INCP et al, RELATION and SPATREL, have already been described. Those which remain are:

FROM1 and FROM2, which are used in MBUILD acts to describe the "old" information from which some "new" information is constructed. These roles take embedded conceptualisations for fillers.

EMOTION, used in MFEEL, and taking one of a small set of primitive tokens representing an emotion felt by somebody.

SOUND, used in SPEAK events, whose filler must be one of a set of tokens representing sounds.

CERTAINTY, which is an optional role in an MLOC state, and which is used to distinguish the English verbs SUSPECT, THINK, KNOW, and BELIEVE. The filler is a number in the range 0 to 1.

MANNER: an embarassment!

The role MANNER may be used in any EVENT, and may take tokens such as QUICKLY, REPEATEDLY; it is an unhappy bedfellow with the other roles, and the suspicion that this is merely a hack to get adverbs in somehow is well founded. One of the most pressing problems with this new implementation of CD is the satisfactory representation of adverbs in general; and it is not anticipated that MANNER will survive when this problem is seriously considered.

2.3.3) FOCUS paths

The ordering of the roles within a CDform is immaterial; the examples given earlier were ordered to simplify the annotation, and were not in the order which is conventionally adopted in the program. The names of the roles serve as labels on substructures of a CDform, and entire CDforms are also labelled with their type. This labelling can be used to construct descriptions of locations within a CDform: such descriptions I call paths. The programs based on the original implementation of CD used similar paths to access substructures; in the present context, where the labelling is augmented with the type of structure, the paths are correspondingly different. For example, in one of the CDforms given earlier to illustrate states,

```
                       "FRED KNEW THAT MARY HAD GONE TO THE SHOPS"
((STATE (STATENAME MLOC)
        (INCP FRED)
        (TIME (NAMED TIMEPOINT1)
              (COMPARISON (BEFORE *NOW*)))
        (MOBJECT (EVENT (ACTOR MARY)
                        (ACT PTRANS)
                        (OBJECT MARY)
                        (FROM DUMMY-PLACE1)
                        (TO SHOPS1)
                        (TIME (NAMED TIMEPOINT2)
                              (COMPARISON (BEFORE *NOW*))
                              (COMPARISON (BEFORE TIMEPOINT1))))))
```

the path (STATE MOBJECT) may be used to pick out the entire embedded conceptualisation, that is, ((EVENT ...)). Similarly,
the path (STATE INCP) leads to (FRED)
the path (STATE MOBJECT EVENT TIME NAMED) leads to (TIMEPOINT2)

One crucial difference between this new implementation of CD and the original implementation lies in the fact that these paths may explicitly appear in the CDform. Paths occurring within the CDform are prefixed by the label FOCUS, and so are called focus-paths.

They are used to embed one conceptualisation within another, in such a way that a particular part of the CDform is "in focus". This is the essential tool which gives the new CD many valuable features. For instance, the examples in figure 2.1, cited earlier as evidence of the non-uniform representation of identical information in Schank's system, may be recast in the new CD as shown in figure 2.2.

--------------------------- Figure 2.2 ---------------------------

```
   "The hat is red"                        "The red hat ..."
((STATE (STATENAME COLOUR)             ......
        (THING HAT1)                    (FOCUS (STATE THING))
        (VAL RED)                       (STATE (STATENAME COLOUR)
        (TIME (NAMED *NOW*))))                 (THING HAT1)
                                               (VAL RED)
                                               (TIME (NAMED *NOW*))))
```

--------------- Uniform representation in the new CD ---------------

It can be seen that the information about the hat's redness is expressed in identical fashion, regardless of context. This I believe to be a desirable property.

This same mechanism can treat deeply-embedded relative clauses: for example, "The banana I asked you to give the man", becomes (omitting detailed time descriptions)

```
(... (FOCUS (EVENT MOBJECT CAUSE ANTECEDENT EVENT OBJECT))
     (EVENT (ACTOR *SPEAKER*)
            (ACT MTRANS)
            (FROMCP *SPEAKER*)
            (TOCP *HEARER*)
            (MOBJECT (CAUSE (ANTECEDENT (EVENT (ACTOR *HEARER*)
                                               (ACT ATRANS)
                                               (OBJECT BANANA1)
                                               (FROM *HEARER*)
                                               (TO MAN1)
                                               (TIME ...)))
                            (RESULT (STATE (STATENAME JOY)
                                           (THING *SPEAKER*)
                                           (VAL (HIGHERBY 2))
                                           (TIME ...)))))
            (TIME ...)))
```

Embedded conceptualisations, prefixed with a focus-path as in these examples, may be used in the same way as simple tokens, to fill roles in larger conceptualisations; the focus-path provides a mechanism for interlacing two CDforms.

Focus-paths are applicable to many situations besides the obvious ones of representing adjectives, relative clauses and possessives. For example, comparative adjectives may be represented by having some dummy value, and focussing on that value within a HIGHERTHAN clause. So,

```
                "FRED IS BIGGER THAN MARY"
((STATE (STATENAME SIZE)
        (THING FRED)
        (TIME (NAMED *NOW*))
        (VAL (HIGHERTHAN (FOCUS (STATE VAL))
                         (STATE (STATENAME SIZE)
                                (THING MARY)
                                (TIME (NAMED *NOW*))
                                (VAL DUMMY-VAL1))))))
```

Similarly, conjoined modifiers or adjective strings may be easily handled:

```
                " ... THE BIG RED LORRY"
(... (FOCUS (STATE THING))
     (STATE (STATENAME SIZE)
            (VAL 4)
            (THING (FOCUS (STATE THING))
                   (STATE (STATENAME COLOUR)
                          (THING LORRY1)
                          (VAL RED)))))
```

2.3.4) Superprimitives

One of the principal attractions of a primitive-based representation language is the concomitant simplicity of the matching procedures: one of the disadvantages, from the point of view of a complete system handling ordinary text, is the explicitness of representation that appears to be necessary. This disadvantage is also apparent when, in the course of producing inferences, it is necessary to inspect the surrounding context.

In Schank's original CD, the role-filler primitives DO, JOY and CONCEPTS were "fuzzy" primitives: the act DO could be matched with any other act; the state JOY could be matched against some other states; and the special token CONCEPTS could match against any whole conceptualisation. Tokens with this fuzzy matching property may be termed superprimitives, indicating not that they are more primitive than primitives, but that they give a (slightly) higher level of description. It may be noted in passing that Wilks used some primitives in this fashion: his '*ANI', for instance, could match 'MAN', 'BEAST', and 'FOLK'.

It has proved useful to extend this idea by introducing a small number of special tokens, each of which can be matched to a small set of proper primitives: for example, **POSEMOTION** may match *LOVE*,*LIKING*, or *ATTRACTION*, and **SOUND** may match *SPEECH*,*LAUGHTER*,*SONG* or *NOISE*. These superprimitives may appear in CDforms in the same fashion as any other primitive, but have special behaviour concerning pattern-matches and the merging of CDforms. This will be discussed further in Chapter 8, section 8.3.2, where the superprimitive classifications are given in full.

In AD-HAC, the state BENEFIT exhibits the fuzzy matching properties shown by JOY in Schank's CD; DO and CONCEPTS also are superprimitives. My notation in this area has become somewhat muddled, however. In general, the names of role-filler primitives begin and end with asterisks: for example, *SPEECH*, *MANY*, *BELOW*, *EYE*; and the names of superprimitives begin and end with doubled asterisks: thus **POSEMOTION**, **SOUND**. However, primitive tokens like ATRANS, MLOC, CAUSE etc., are also primitives, yet their names do not have the asterisks; and, as just noted, the tokens DO, BENEFIT and CONCEPTS are superprimitives, but lack the double asterisks. This is of no real consequence, but the muddle must be admitted.

2.3.5) Time representation

There are two sorts of time token used in the system, TIMEPOINTs and TIMESPANs. Each TIMESPAN indicates an interval of time, and is bounded by two TIMEPOINTs, indicated by TS (Time of Starting) and TF (Time of Finishing). A TIMEPOINT indicates a point in time. The tokens *NOW*, *ABINITIO* and *EVERMORE* are special cases of TIMEPOINTs, corresponding to now, infinitely long ago and infinitely far ahead. If a TIMESPAN lacks either (or both) TS or TF, these are assumed. As earlier examples show, the time of a conceptual fragment is indicated by the label TIME, and the name of the time token is labelled NAMED. Examples of TIMESPANs are:

```
(TIME (NAMED TIMESPAN1)    or, equivalently,   (TIME (NAMED TIMESPAN1)
      (TS *NOW*))                                    (TS (NAMED *NOW*)))

(TIME (NAMED TIMESPAN2)
      (TS TIMEPOINT1)
      (TF TIMEPOINT2))
```

If it is necessary to compare two time tokens, a specification of the relationship is labelled with COMPARISON. The specification itself consists of a time token, or time specification, labelled with one of the <u>time relation labels</u>, which are

BEFORE , AFTER , NOTBEFORE, NOTAFTER , DURING , -DURING

For example, to indicate that TIMEPOINT3 lies within TIMESPAN2, we say

```
(TIME (NAMED TIMEPOINT3)
      (COMPARISON (DURING TIMESPAN2)))
```

Some more complex examples are given below.

```
(TIME (NAMED TIMEPOINT3)
      (COMPARISON (DURING (NAMED TIMESPAN2)
                          (TS (NAMED TIMEPOINT1))
                          (TF (NAMED TIMEPOINT2)
                              (COMPARISON (BEFORE *NOW*))))))

(TIME (NAMED TIMESPAN3)
      (COMPARISON (-DURING (NAMED TIMEPOINT2)
                           (COMPARISON (AFTER *ABINITIO*)))))
```

The time relation label DURING can only be used to relate some time token to a TIMESPAN: in the first example given above, TIMEPOINT3 is being related to TIMESPAN2, which has a TS and a TF which are not shown but which are held on its property list. Suppose these are TIMEPOINT1 and TIMEPOINT2 respectively: the time relationships which hold may be diagrammed as follows:

```
----------> time's arrow goes from left to right ------>

           TIMESPAN3
           |       |
      (TS) |  (TF) |
           |       |
  TIMEPOINT1       TIMEPOINT2
           |       |
           |-?-|-?-|
               |
               |
          TIMEPOINT3
```

The relation DURING refers to the TS and TS of the TIMESPAN, and indicates that the NAMED time lies somewhere between these points. This is expressible in terms of the simpler time relations, BEFORE and AFTER. Since however it may be the case that TIMEPOINT3 is actually identical with one of these points, the simple relations NOTBEFORE and NOTAFTER are actually used.

The second example shown above differs from the first in two respects: firstly, the TIMESPAN with which comparison is being made has its end-points explicitly given; secondly, one of these it itself compared with another time token, viz. *NOW*.

The third example shows the use of the inverse label, -DURING. This label requires the NAMED time token to be a TIMESPAN, and indicates that that span includes the compared token.

The labels BEFORE and AFTER are used liberally; indeed, the annotated examples of CDforms have illustrated their use, and nothing more will be said about them here.

This may seem like an elaborate treatment of time; yet in fact it is not even adequate for the representation of many temporal references which occur in real texts. For instance, there is no way in which "FRIDAY EVENING" or "3 pm" can be referred to. This is one of the outstanding problems which must be tackled in future.

2.3.6) Features of tokens, and DUMMY tokens

In Schank's CD, concrete nouns were represented by means of picture producers: these were actually Lisp atoms, on whose property lists were stored various forms of information; the printed versions of CD structures expanded these property lists into the form shown in earlier in figure 2.1. In these printed versions, there appeared tokens like £PHYSOBJ and £HUMAN, which indicated that the picture producer was an instance of the class £PHYSOBJ etc.

In this new version of CD, concrete nouns are represented by Lisp atoms, usually with mnemonic names. Thus, tokens like HAT3, TABLE1 and JOHN appear in the printed CDforms. These tokens are created behind the scenes, and are equipped with features which serve to classify the tokens. Thus, suppose the noun TABLE occurs in an input text: the analyser will create a new atom, say TABLE1, which will be used to represent this concept; it will also record, on its property list, that TABLE1 ISA TABLE.

The features of TABLE1 may be found by following the ISA link to TABLE, and looking at its "features" property: this will say "(RIGID)".

The features are organised in a hierarchy, shown below: thus, if TABLE1 has the feature RIGID, then it has all the features
(RIGID SOLID CONCRETE ANYTHING ABSOLUTELYANYTHING)

```
              ____ABSOLUTELYANYTHING__________
             /     /      |     \             \
            /     /       |      \             \
   ABSTRACT  DIRECTION  MESSAGE  AGENCY      _____ANYTHING_____
      / \                                   /         |        \
     /   \                                 /          |         \
 GERUND ABSTRACTNOUN          ______CONCRETE_____   ORGANISATION   PLACE
                             /                  \                  / \
                            /                    \                /   \
                    ___SOLID___                  FLUID  OPEN-PLACE ENCLOSED-PLACE
                   /     |     \                 / | \
                  /      |      \               /  |  \
            FLOPPY  ANIMATE     RIGID     POWDER  GAS  LIQUID
                      / \        / \
                     /   \      /   \
              BEAST PLANT INERT MACHINE
               / \
              /   \
          HUMAN ANIMAL
           / \
          /   \
       MALE  FEMALE
```

These features are called primary features, and they are used throughout the system to give information about the tokens to which they are (indirectly) attached. There are also "secondary features", such as PROPERNAME or MASSY, which give purely linguistic information about the associated words.

Each primary feature has associated with it a dummy. This is a token such as DUMMY-HUMAN or DUMMY-MALE, which can be used to represent an prototypical bearer of that feature. Instances of these dummy tokens, particularly tokens such as DUMMY-MALE1, DUMMY-MALE2, are created by the analyser to represent pronouns.

Each primary feature also has an inverse feature, which may be explicitly given for particular token types. Thus, the token which the analyser creates for the pronoun THEY will be say DUMMY-UNKNOWNS3, and this will lead to the feature set (ANYTHING NOTFLUID); this enables the inference mechanism to match such a token against almost anything, but not to match it against tokens bearing any of the properties
FLUID, LIQUID, POWDER, GAS.

2.4) Criticisms of the AD-HAC development of CD

One wholly spurious criticism concerns the use of tokens such as HAT3, JOHN, TABLE1. Tokens like this, bearing such an obvious resemblance to English words, are often thought to be just English words; it tends to be assumed that, because a token is printed as "HAT", the only knowledge that a system has is bound up in this sequence of characters. This is patently false. The only reason for using mnemonic tokens is to assist people who inspect the programs internal data structures. Indeed, there is an option in AD-HAC which permits all these generated tokens to be Lisp GENSYMs: the program actually works better when this option is set, being faster and using less space; but if I were to explain its representation language using examples generated in this mode, I would constantly be saying "'G0017' represents John, 'G0023' represents the banana". Using perspicuous tokens does not mean that the name is being taken for the meaning.

As mentioned in section 2.3.6 above, tokens have associated features, and these serve to convey some of the meaning of nouns. All the concrete nouns known to the system are also placed on an ISA hierarchy, which functions as a rudimentary thesaurus. Additionally, many of them have functions; these are little used by the sentence analyser, but serve rather to index object-specific knowledge for use by the inference mechanism: this knowledge takes the form of inference networks, which can be activated when tokens bearing these function labels occur in contexts where this knowledge is appropriate.

There are however some more fundamental weaknesses in this language, specifically in its treatment of quantification and of spatial relationships. Attempts have been made to cover both topics. Quantification is represented by a STATE, named QUANTIFY, which has special properties relating to the scope of information expressed within that state. Thus

```
(... (FOCUS (STATE THING))        ; "... all happy men"
     (STATE (STATENAME QUANTIFY)
            (VAL *EVERY*)
            (THING (FOCUS (STATE THING))
                   (STATE (STATENAME JOY)
                          (THING MEN1)
                          (VAL 2)))))

(... (FOCUS (STATE THING))        ; "... all men -
     (STATE (STATENAME JOY)       ;      who incidentally are happy -"
            (VAL 2)
            (THING (FOCUS (STATE THING))
                   (STATE (STATENAME QUANTIFY)
                          (THING MEN1)
                          (VAL *EVERY*)))))
```

are not equivalent statements, and are handled appropriately by all components of the system being described. Note that, if QUANTIFY and *EVERY* were replaced with, say, SIZE and -3, the two fragments would refer to "SMALL HAPPY MEN" and "HAPPY SMALL MEN", and would be equivalent: the special status of the state QUANTIFY lies in its special scoping effect.

Spatial relationships are represented by means of the role SPATREL which may appear in the state LOC, and may have values such as *ON*, *BELOW* etc. However, no mechanism exists for the representation of distance between points in space. The problem here is similar to the representation of duration of time spans. In both cases, a very precise measure is available, but usually a sloppy measure is required. A further extension of the idea of superprimitives may suffice, but it appears that the number of such primitives would be enormous.

Another problem is that there is no means within the representation language to distinguish between restrictive and non-restrictive relative clauses; both are reflected by the use of focus-paths surrounding embedded CDforms. While this uniformity of representation has some definite merit, it would be nice to have some way of telling whether the embedded CDform is intended to restrict the referents of the token in focus, or intended merely to further qualify the token.

To summarise, the new implementation of CD represents a significant improvement over the original, largely because the use of focus-paths within CDforms permits canonical representation; but also because the representation of time is more thorough, and because of the introduction of several superprimitives which allow a degree of slop. Nevertheless, there remain flaws and weaknesses, most particularly in the representation of space, time and quantification.

There are other problems which have become apparent as AD-HAC grew: these later problems are discussed in the concluding chapter.

2.5) Implications of this representational language

For the analyser, a language such as this implies that the same, or at least equivalent, structures of semantic primitives must be built for all different acceptable ways of expressing the same thought. The constraints on the CDforms, such as the interpretation of ABILITY relating to state-based CDforms, must be particularly observed and exploited. The addition of FOCUS paths to the structures relating to relative clauses, adjectival modifiers and all other forms of modification, must be effected consistently. A major problem in analysis is the interpretation of tensing patterns, both in complex sentences and between sentences. This point will be discussed further in Chapter 5, section 5.3.5.

The primitive acts and statenames provide a natural organisation for an inference processor. The presence of superprimitives, as mentioned above, facilitates the processes of matching propositions and of interrogating context, at the (small) price of providing special mechanisms for merging propositions. For technical reasons concerning space and search efficiency, the information passed between the high-level components of the system, ie the analyser, generator and inferencer, are reconfigured within the inferencer to property-list format; and so there are additional mechanisms for translating in both directions. The presence of features, ISA links etc must also be utilised, both when attempting to match CDforms and when determining which inferences to make.

For an English generator, the complete absence of syntactic information coupled with the presence of primitives rather than English words implies that the generator itself must make decisions about word usages, syntactic frameworks, and the level of specification of items occurring in its inputs.

2.6) Wilks's semantic-primitive representation scheme.

A completely different representation language was developed by Wilks, for use in his machine translation project. This language was also based on the use of semantic primitives.

The most attractive feature of this language was the flexibility with which primitives could be combined into formulas to represent the senses of individual words. For example, the (single) formula for the word "policeman" was

((FOLK SOUR) ((((NOTGOOD MAN) OBJE) PICK) (SUBJ MAN)))

ie. somebody who selects a bad person from amongst a group
This is a very simple example of the combination of primitives into formulas. Some more examples, and more explanation, is given in Chapter 4.

Wilks used some 80-100 primitives, though in Wilks's sense the case labels (like SUBJ, OBJE and SOUR in the above example) were counted as primitives: Schank, on the other hand, did not consider the names of cases to be primitives. Wilks further notes that the set of primitives which he uses corresponds well with the hundred or so words which are most frequently used in the definitions of other words in a dictionary; and adduces this as evidence that his is a "good" set of primitives, meanwhile arguing [Wilks, 1977] that there is no reason to seek "the right" set.

Though Wilks uses primitive terms to good effect in characterising the meanings of words, the larger structures in which these word-sense formulas are embedded are less uniform and less natural than those of CD. The formulas are assembled in triples, called <u>templates</u>, which are themselves linked by <u>paraplates</u>.

The templates may in many cases be thought of as actor-action-object triples; other situations which are easily handled by templates include, for example, thing-be-adjective. The use of "dummies" permits intransitive verbs to be represented, by simply omitting the "object" slot of the template; but difficulties arise in circumstances where more than two noun phrases are associated with a verb - as happens, for instance, with verbs which expect an indirect object. For these cases, paraplates are employed to tie several templates together into a larger structure. Paraplates are usually associated with prepositions in Wilks's system.

When one considers that many of Schank's primitive acts require the specification of many cases, it becomes apparent that tying templates into larger structures is an inelegant solution to a widespread problem. The basis of the trouble, I believe, is that the actor-action-object paradigm which led to the three-slot nature of templates is not sufficiently general, and that extra generality had to be grafted on by joining these triples together.

Chapter 3: Generation

This chapter describes the sentence generator used in AD-HAC. The basic components of the system were developed in what may appear to be a rather idiosyncratic order: first the generator, then the analyser, and finally the inference mechanism. The reason for this was quite simple. It was immediately apparent that the representation adopted was of critical importance; that some of the details of representation could not be determined outside the context of some concrete program; and that changes to the representation would entail modification of that program. This indicated that the least complex system component utilising the representation should be attempted first. In AD-HAC, the least complex of the components is the generator, which, as will become apparent, is based largely upon Goldman's previous work on BABEL.

This chapter on generation appears here for several reasons. Firstly, the historical development of the components as a whole may be reflected in the order of their discussion; secondly, the fact that this generator is based upon previous work, and shares much of the apparatus developed by Goldman, deprives this chapter of much claim to originality. Finally, and relatedly, much of the discussion may be familiar and hence simpler to follow than that in succeeding chapters; for readers acquainted with this discussion, more familiarity with the representational system described in the preceding chapter may be gained.

The overall problem of sentence generation may be characterised as having three parts. The first of these, choosing words, has, in its more demanding aspects, often been neglected. In a rigorous treatment this extends from the choice of verbs, nouns and adjectives, through the choice of function words, such as prepositions and conjunctions, and may be extended to the determination of those contexts in which the use of a pronoun is appropriate.

The second part, word formation, passes under the name of "morphology". This is one of the fundamental concerns of linguistics, but has seldom been considered a serious problem in computer-based language processing systems outside the MT context; for the range of vocabulary with which systems not attempting translation are endowed, a superficial treatment will often suffice.

The third part of the problem, word ordering, is often regarded as the principal aspect of the problem of language production. The study of word ordering has occupied many linguists, and their researches have taken many forms and produced many theories. Foremost amongst these theories in recent decades is the theory of Transformational Grammar.

Section 3.1 comments briefly upon some of the earlier work on generation of natural language; Section 3.2 provides an overview of Goldman's BABEL, as a prelude to section 3.3, which provides an overview of AD-HAC's generator. Section 3.4 provides more detail about this generator, and section 3.5 indicates those areas in which further work would have to be done.

3.1) Previous work on generation

Generation has traditionally been the poor relation in NL work: far more effort has been directed towards the analysis of language. Nevertheless, generators have been written, either for their own sake, or as part of some larger system. The following paragraphs sketch some of these earlier generation programs.

3.1.1) Friedman

Friedman's work was concerned with transformational generative grammar, and aimed to provide a testbed for exploring the effects of new transformations and constraints. No effort was made to ensure that the program's output was semantically sensible: the program had nothing to say, but knew lots of ways of saying it.

3.1.2) Winograd

Winograd was concerned primarily with analysis and various forms of inference [Winograd, 1971]. Though his program did indeed have reason to communicate, the range of expression needed was very small. Winograd made no comprehensive attempt to generate English from any meaning representation: much of the generator's output took the form of "patterned responses", involving only the insertion of content words into prepared templates. There was a rudimentary ability to construct descriptive noun phrases, and also to incorporate these into clauses formed by programs attached to the "concepts" of the blocks world. A small number of discourse heuristics were also employed to render the generator's output reasonably fluent.

3.1.3) Herskovits

Herskovits constructed the French-generation component for the English-French translation program written by Wilks, which employed Preference Semantics as its meaning representation [Herskovits, 1973]. Intermingled with this representation was information, derived directly from the surface words of the English input, which gave the appropriate French words, and often specific constructions, to be used in the French output. This information was held in "stereotypes" which were associated with individual English word senses. Whilst this approach can produce fair translations - and could no doubt provide truly excellent "paraphrases" - it would be difficult to adapt to the task of expressing inferences, for which new word senses would have to be discovered by other means.

3.1.4) McDonald

McDonald's MUMBLE program is intended both as a useful generation program, and as a psycholinguistic model [McDonald, 1977]. It is constrained to produce sentences in a deterministic left-to-right fashion; one of its more interesting consequent properties is that, like people, it sometimes does make wrong decisions and finds itself unable to complete a sentence it has started. Unlike any other generation program of which I am aware, MUMBLE takes account of rhetorical factors, and uses these to determine how best the speaker's intentions for the text may be effected.

3.1.5) Goldman

Goldman's BABEL was used in the MARGIE program [Goldman, 1974]. It accepted conceptualisations from either the analyser or the memory/inference component, and cast these into English. Since one of the aims of the MARGIE project was to prove that CD was capable of capturing meaning, BABEL was designed to produce many synonymous English sentences for each input conceptualisation. Another aim of the overall project was to demonstrate that CD could be used as an interlingua, and so Goldman produced a modified version which produced German sentences. Since then the basic program has been adapted to produce a wide variety of languages, apparently without significant development.

3.2) Overview of BABEL

As just noted, Goldman's BABEL produced English sentences to reflect the meaning of conceptualisations. This represented the first attempt to generate sentences given some representation of their desired meaning; and since this representation was very close to that used in AD-HAC, BABEL's approach to generation provided a relevant model on which a new effort could be based. The following sections describe AD-HAC's generator, and give special emphasis to the improvements and developments embodied in it. I wish to give here a brief overview of BABEL's operation, so that subsequent description can rest on an understanding of how pieces of AD-HAC's generator are fitted together to make the whole; and so that the similarities with, and differences from, BABEL may be easily pointed out.

The structures from which BABEL generated a sentence contained no reference at all to the main verb to be used, but did contain a CD expression of the desired meaning of the sentence, and, indirectly, reference to the specific nouns to be used in the sentence. The first operation to be performed then was to select a suitable main verb: this was done by the use of discrimination nets. The discrimination nets applied performed various tests on the conceptualisation, and ended up by selecting a suitable case frame. The case-frame contained, besides a verb stem, a set of "syntactic roles" for noun groups and subordinate clauses, each being associated with a specification of a place within the conceptualisation which should be used to fill these syntactic roles. The case-frame thus contained sufficient information to unpick this conceptualisation in order to associate particular conceptual roles with particular syntactic roles.

The verb stem and all the syntactic labels were placed in a newly-created node in a "syntax net": subordinate clauses were treated by a recursive application of the same mechanisms, and the nodes for these clauses were linked to the main clause node via the appropriate syntax label. Noun groups corresponded to "picture producers" (PPs), and caused a similar linking of syntax nodes via syntactic role labels. A simple mechanism permitted the attachment of prepositions to these nodes at the same time.

The resulting syntax net was scanned by a simple ATN grammar, which used routines to effect tensing, agreement between subject and verb, and the location of irregular verb forms in a lexicon. The operation of the grammar upon the syntax net caused the production of the actual text sentence.

The overall structure of BABEL was as diagrammed below. The "concexicon" was the external file in which the case-frames were stored.

```
|Discrimination |      | Concexicon |       | Property |
|    nets       |<--         ^         -->|  scales  |
                   \         |         /
                    \________|________/
  / Memory \       / Generation       \        / Grammar   \
 |  Model   |<---->|  control          |---->|  control     |
  \________/       \  algorithm       /       \ algorithm  /
                            |                    |        |
                            v                    v        v
                   /  Language        \    | Lexicon | | AFSTN   |
                  |   specific         |             | grammar |
                   \  functions       /
```

3.3) Overview of AD-HAC's generator

Much of AD-HAC's generator is the same as BABEL, but there are some differences, partly because the representation languages are not identical, and partly because improvements on BABEL's performance could be achieved by modifying the approach.

Section 3.3.1 describes in some detail those aspects of the present generator which are essentially copied from BABEL. Section 3.3.2 then discusses how, and why, the input CDforms are processed before the generator proper goes to work; this process is called canonicalisation, and is performed by the inference mechanism also. Section 3.3.3 shows that discrimination nets may usefully be permitted to execute actions, rather than being restricted to merely conducting tests; and shows that for some purposes it is necessary to have a collection of discrimination nets, rather than just one huge net. Section 3.3.4 describes the greatest difference between BABEL and the present generator, which lies in the way the syntax net is used to produce the final text sentence: briefly, where BABEL used an ATN grammar to inspect the net, in AD-HAC the syntax relations correspond to the names of small programs, and there is a standard and uniform ordering which can be applied to these roles which will produce grammatical English sentences.

3.3.1) Similarities with BABEL

Since, like BABEL, my generator is given no information about the words to be used in the generated sentence, but only a conceptual specification of meaning, it must select a set of words. For simple nouns, this is easy since a token like LAMP-POST1 has a ISA link pointing to the "concept" LAMP-POST, which in turn has an ENGLISH property pointing to the word list (LAMP POST); in the case of a token like DOG1, whose ISA link points to the concept "DOG", the homography of the concept and the appropriate English word can be exploited to remove the need for even the ENGLISH property.

For verbs, however, there is no simple mechanism like this available, because the conceptual pattern determines the meaning of a conceptualisation, and the verb is the most important element for conveying meaning. (The meaning of abstract nouns is related to this; for instance, the pattern which corresponds to 'DECIDE' will also correspond to

'DECISION') So some mechanism for inspecting the pattern of a conceptualisation must be used: for this task, like Goldman, I use discrimination nets; and these yield a reference to a case-frame. A simple example of such a case-frame is

```
(TELL1                  : the key for lookup purposes
   TELL                 : the verb stem
   ((ACTSBJ             : a syntactic label, for the subject
      (EVENT ACTOR))    : specifying a location in the conceptualisation
    (OBJ                : syntactic label, for the object
      (EVENT TOCP))     : another location
    (S2                 : syntactic label, clause preceded by 'THAT'
      (EVENT MOBJECT)))) : location of a sub-conceptualisation
```

When such a case-frame has been chosen by the discrimination nets, the verb stem is added to a node in a new structure, the "syntax net". This node will correspond to a clause in the sentence. For the moment I shall assume this is the main clause, and so I shall refer to the created node as the "main clause node". Miscellaneous items, such as negation, tense, form and modality will be added at this point. Let us assume the main clause node is Syntax-node-1; then the syntax net at this point will consist of that one node, which will contain

```
((LEXVERB . TELL)
 (TENSE . PAST)
 (FORM . NIL))
```

if we assume no negation, no modality and past tense.

Attention now shifts to the other components of the case-frame. These are pairs of syntactic labels and "path specifications", optionally followed by specifications of prepositions and the like, which this simple example lacks. With a few exceptions not treated here, the syntactic labels fall into two categories corresponding to noun phrases and subordinate clauses. When a label corresponding to a subordinate clause is found, like S2 in this example, a new node is created; the specified sub-conceptualisation, identified by the path specification, is presented to the discrimination nets, and a verb-sense is found and processed in the same manner as the main clause. Upon completion of this process, the node for the subordinate clause is linked to the main-clause-node, prefixed by the syntactic function label in the original case-frame.

Labels corresponding to noun phrases also cause the creation of new nodes; this process is quite complex and is dealt with in section 3.4 below. For the moment, all that matters is that an appropriate node is also linked, via the syntactic label, to the clause node.

During the process of case-frame interpretation, several subsidiary processes must occur: appropriate tenses must be determined, prepositions may be needed, subconceptualisations may need to be preprocessed before verb senses are selected, and some specific transformations of the syntax net may have to be applied; these are all discussed in the next section.The result of this phase of processing is a network whose nodes contain information about specific words to appear in the sentence, and information about the syntactic relationships between items. An example is given here. (The sentence produced is "JOHN TOLD A CHILD THAT A MONKEY HAD MANY BANANAS.")

<u>Input conceptualisation</u>

```
((EVENT (ACTOR HUMAN-JOHN)
        (ACT MTRANS)
        (MOBJECT
            (STATE (STATENAME POSS)
                   (THING (FOCUS (STATE THING))
                          (STATE (STATENAME QUANTIFY)
                                 (THING BANANAS1)
                                 (VAL *MANY*)
                                 (TIME (NAMED TIMEPOINT1)
                                       (COMPARISON (BEFORE *NOW*)))))
                   (VAL MONKEY1)
                   (TIME (NAMED TIMEPOINT1)
                         (COMPARISON (BEFORE *NOW*)))))
        (FROMCP HUMAN-JOHN)
        (TOCP CHILD1)
        (TIME (NAMED TIMEPOINT1)
              (COMPARISON (BEFORE *NOW*)))))
```

<u>Map of syntax net</u> (The top node is Syntax-node-1)

```
Syntax-node-1 contains:  ((ACTSBJ . Syntax-node-2)
                          (TENSE . PAST)
                          (FORM)
                          (LEXVERB . TELL)
                          (OBJ . Syntax-node-5)
                          (S2 . Syntax-node-8))

Syntax-node-2 contains:  ((LEXNOUN . Syntax-node-3))

Syntax-node-3 contains:  ((ENGLISH . Syntax-node-4))

Syntax-node-4 contains:  (JOHN)

Syntax-node-5 contains:  ((LEXNOUN . Syntax-node-6))

Syntax-node-6 contains:  ((ENGLISH . Syntax-node-7))

Syntax-node-7 contains:  (CHILD)

Syntax-node-8 contains:  ((ACTSBJ . Syntax-node-9)
                          (TENSE . PAST)
                          (FORM)
                          (LEXVERB . HAVE)
                          (OBJ . Syntax-node-12))

Syntax-node-9 contains:  ((LEXNOUN . Syntax-node-10))

Syntax-node-10 contains: ((ENGLISH . Syntax-node-11))

Syntax-node-11 contains: (MONKEY)

Syntax-node-12 contains: ((LEXNOUN . Syntax-node-13))

Syntax-node-13 contains: ((QUANTIFIER . MANY)
                          (ENGLISH . Syntax-node-14))

Syntax-node-14 contains: (BANANAS)
```

In this example, the verb sense chosen to express the conceptualisation was TELL1, whose concexicon-entry reads:

```
(TELL1 TELL
       ((ACTSBJ (EVENT ACTOR))
        (OBJ    (EVENT TOCP))
        (S2     (EVENT MOBJECT))))
```

For this verb sense, the verb stem is TELL, and the syntactic roles required are ACTSBJ (the subject), OBJ (the object), and S2 (an embedded clause preceded by "THAT"). Each role name is followed by a "path specification", which is interpreted by taking the first member of the list, and treating the current conceptualisation as an association list; the part of the conceptualisation keyed by this item is then the context for the next member of the specification. Thus, in the example,

(EVENT ACTOR)	points to	(HUMAN-JOHN),
(EVENT TOCP)	points to	(CHILD1)
and (EVENT MOBJECT)	points to	((STATE (STATENAME etc...)))

The syntax role S2, for the embedded clause, again requires the selection of a verb sense. The discrimination nets are used to find the sense HAVE1, whose concexicon-entry is

```
(HAVE1 HAVE
       ((ACTSBJ (STATE VAL))
        (OBJ    (STATE THING))))
```

As can be seen in the above example, the roles specified by the verb-sense framework are reflected in the current syntax-node, and indicate another syntax-node which contains further information. In this example, the element (ACTSBJ (EVENT ACTOR)) in the case-frame for TELL1 is reflected as (ACTSBJ . Syntax-node-2) in node 1 of the syntax net. Also, miscellaneous relations are added for TENSE, FORM (here always null) and LEXVERB.

This much of the generator's operation is essentially the same as BABEL. One further close similarity remains, the provision of a lexicon which contains irregular verb forms. Normally the present, past and participial forms of verbs are computed by appending -s,-d or -ed, -ing to the verb stem. Forms which cannot be so computed are held in the lexicon: thus, for the present example, TELL -> TOLD.

3.3.2) Canonicalisation

The first thing the generator does is to "canonicalise" the conceptual structure. This is done for two reasons: to collect together all the modifiers of nominals, making them easily accessible to the noun-phrase generation routines; and building a data structure which permits easy determination of temporal relationships.

All the modifiers, ie conceptual structures headed by FOCUS, are collected on a list called "global-subpropositions", which is indexed according to the modified nominal, and organised in such a fashion that a unique copy of each modifier is held. Quantification is specially treated at this point - quantified structures are not dissected: thus modifiers within a quantified structure have no effect outside the scope of that quantifier. All the information in "global-subpropositions" is then plugged into the main conceptualisation, so that this contains all available information.

The information collected in this fashion is used principally for the production of noun phrases, but is sometimes also useful for verb-selection. Occasionally, a verb-choice will be dependent upon the presence of some modifying structure, and in such a case it is often useful to be able to delete this modifying structure by specifying the general pattern which it will have. For example, one of the case-frames for MEET is "X meet Y by Z"; but, unless part of the information is deleted at some stage, we will get odd sentences like "JOHN MET MARY, WHO WAS NEAR A CHURCH, BY THE CHURCH".

3.3.3) Usage of discrimination nets

The general application of discrimination nets in AD-HAC is similar to BABEL; however there are several differences in detail. Firstly, in some circumstances the application of a discrimination net may be entirely circumvented. This will occur when the generator is working at the behest of the inference component, and the latter can determine at least the top level construction appropriate (eg IF ... THEN ...). This parallels modifications to BABEL for use by SAM, PAM, QUALM etc.

Secondly, the discrimination nets are not constrained to apply tests only, but are permitted to perform certain actions. This enables the discrimination nets to perform such tasks as adding query-markers,neg-markers and modals to the syntax net, and also to specify the deletion of parts of the current structure. These are operations which, in BABEL, could be done only within word-sense processing. An unpleasant consequence of this restriction was that all operations of this sort had to be done by a case-frame; and therefore a "word-sense" had to be provided. Many of these operations corresponded in practice to the modal structures: BABEL was unable to use modals except through the device of having "word-senses" like ABLE1, OUGHT1. These introduced a non-uniformity into the structure of the syntax nets, which is eliminated in AD-HAC's generator as a direct consequence of permitting the discrimination nets to perform these actions directly.

A prerequisite for this style is that the discrimination nets be segmented, and that one net may specify that the current conceptualisation (or part of it) should be examined by another net. For example, the representation of OUGHT TO is a causal, where the antecedent is the action which ought to be done, and the result is the (general) benefit which will ensue. When this pattern is detected, the OUGHT modal can be added to the syntax net, and the antecedent part can be passed directly to the EVENT discrimination net. The result of this is that no dummy node is needed, and the uniformity of syntax-nodes corresponding to clauses can be retained.

Additionally, some new discriminating tests have been provided. The most important of these tests whether some sub-conceptualisation will be expressed using a verb or a conjunction. Such subpart tests are significant for two reasons: firstly, they could be extended to allow abstract nouns to be selected; secondly, they provide a mechanism for making high-level decisions dependent upon their low-level consequences. This is implemented simply by recursively applying discrimination nets to the indicated subpart, and inspecting the structure of the resulting word-sense.

The remaining novel tests simply see whether there is focus on a specified part of the conceptualisation, or whether a specified part will be expressed as a dummy, ie "something" or "somebody". The former allows the choice of verbs which will convey the desired focus, whilst the second permits the selection of word-senses,corresponding to case-frames, which entirely omit stylistically superfluous parts of the conceptualisation. For example, this permits the generator to produce a sentence like "JOHN LEFT THE HOUSE" rather than "JOHN WENT FROM THE HOUSE TO SOMEWHERE".

3.3.4) Interpreting the syntax net

To produce a sentence from a syntax-net, BABEL uses an ATN. The mechanism used here is radically different, in that the syntax net is regarded as a program which can be executed to produce a sentence. Each syntactic role is the name of a function, taking as argument either a syntax node, or a word or special label. The contents of the top node are evaluated, and cause the contents of all other nodes to be either evaluated, or added directly to the sentence (as for Syntax-node-4 above). This mechanism, though very simple, is adequate for the production of quite complex sentences. The only complication is, how does one know the order in which the contents of a node should be executed? It turns out that the answer to this is very simple also: there exists a standard ordering, which, applied to the syntactic functions which may exist on any one node, will yield the correct sequence. If this ordering is imposed when a node is completed - ie a clause has been processed, or a noun phrase - operations such as wh-movement present the only complication; and even this is very much easier than with an ATN.

In this scheme, many operations are facilitated; specifically, the addition of neg-particles, the preposition of auxiliaries for yes/no questions, determiner selection and pronominalisation, and the addition of commas, can be easily programmed in this environment.

3.4) More detail

This section gives more detail about the generator's treatment of two large topic areas: the relationship between case-frames and nodes in the syntax net corresponding to verbs, and the production of noun phrases.

3.4.1) Processing verb senses

3.4.1.1) Tense, form and modality.

Whenever a conceptualisation is processed by the discrimination nets, a verb-sense or a conjunction is selected as the main component of the corresponding clause. Except when a conjunction is selected, it is necessary to select a tense for the corresponding clause. The selection of tense is discussed at length by Goldman, but a need for additional heuristics soon became apparent when the generator began to process "real" conceptual structures produced by the sentence analyser. Furthermore, BABEL did not handle intervals of time; their satisfactory treatment, necessary if the generator is to be reliable as an interpreter of conceptual structures produced by the other components of the system, necessitated an overhaul of the processes of tense and form selection.

BABEL had nine tenses: PAST, PRES, FUT were the simple tenses, and could be joined (for embedded clauses) to produce

PASTPAST PRESPAST FUTPAST PASTFUT PRESFUT FUTFUT

(where FUTPAST is FUT in a PAST environment: eg WILL HAVE DONE)

It appears that the tense FUTFUT, though valid in conceptual terms, does not differ from FUT, in English anyway. Therefore FUTFUT has been discarded.The tenses PRESPAST and PRESFUT are also discarded, since they correspond to the addition of progressive form rather than to a discrete tense structure.

The principal heuristic employed by Goldman was that the tensing structure be derived from the relationship between the TIME of the clause and the TIME of the embedding clause, or the time of utterance in the case of the main clause. This remains valid, given a generous interpretation of time relationships involving intervals, and allowing for the fact that the state IDENTITY does not carry any time information at all.

The absence of progressive form for STATE-based verbs (eg KNOW, OWN) was noted by Goldman, and a problem area in connection with perception verbs (SEE, HEAR, SMELL etc) was also identified. Such verbs I now treat as exceptions, in that they are based on MTRANS, an ACT, but cannot take progressive form merely because of time relationships. A further class of exceptions has been identified, namely verbs based on the act MFEEL; anomalous sentences such as "John thought that Mary was hating him" are not produced.

These exceptional cases, STATE-,MFEEL-based and perception verbs, merely prohibit the addition of progressive form on the basis of time relationships: progressive form is still permitted when the relevant clause is in certain subordinate relationships which demand progressive form.

The form of a verb phrase may have a value from the set {NIL,PROG,BASE,TO}. Normally it will be NIL, but, as discussed above, time-relationship considerations may select PROG. Additionally, certain syntactic roles for embedded clauses may dictate the use of PROG, BASE or TO. These are mutually exclusive. PROG corresponds to progressive form, TO corresponds to the infinitive, and BASE corresponds to the uninflected verb;as in "I saw Fred punch her". Certain syntactic roles, namely INF, PRSNT, GOBJ and their variants, dictate the use of a particular form - TO, BASE and PROG respectively. The functions to which these syntactic roles correspond modify the subordinate-clause syntax node before executing it. The ease of this operation is one of the benefits of abandoning the idea of a general grammar in favour of a functional representation of syntax.

Modality is given a rather extended meaning. Thus the production of words like OUGHT, CAN, BEGAN is specified by "modality", and is often crucially dependent upon the ability to switch between different discrimination nets as mentioned above. Since these "modals" are not necessarily mutually exclusive, the values may be added one-by-one to the syntax net. The function "MODALITY" is an exception to the general rule that the syntactic-role functions take a node as argument.

3.4.1.2) Optional case-frame components

BABEL's verb-senses specified a verb stem and an accompanying case frame, together with miscellaneous operations upon the current conceptualisation such as addition and deletion. Whilst the basic pattern has been retained in AD-HAC's generator, one simple modification has been introduced. Certain elements of the case frame may be specified as optional. Noun phrases, or more commonly prepositional phrases, if optional, will be omitted if they are "dummy", ie they would be expressed as "somebody","something" etc, they do not already appear elsewhere in the sentence, and they are not the focus of the conceptualisation. Similarly, optional verb phrases will be omitted if they would only produce a phrase like "by doing something". This simple mechanism strikingly enhances the quality of the English produced, without either introducing trivially different verb-senses or increasing the complexity of the discrimination nets.

3.4.1.3) Wordsense-specific operations

The word-sense entries in BABEL's "concexicon" may specify operations to be performed upon the current conceptualisation before further processing takes place,and were, in practice, only addition and deletion of parts of the conceptualisation. This ability is retained in AD-HAC, but, since the discrimination nets may themselves alter the conceptualisation, this facility is little used. It has been found desirable to allow certain word-senses to specify operations which may alter the syntax-net which they build. This process I call postediting, and is used principally by the sense of "BE" corresponding to the state IDENTITY, and permits the construction of syntax nets which will produce sentences like:

-IT WAS BILL WHO WENT TO PARIS.

-THE THING I GAVE BILL WAS THE SAME THING WHICH FRED GAVE TO ME.

These cannot be produced by the normal mechanisms, because

(a) "IT" here has no referent, but is an idiomatic usage

(b) "SAME" is functioning syntactically as an adjective, but there is no conceptual correlate other than the IDENTITY which has caused production of the whole sentence. This is one instance where superfluity is desirable.

The other use is for sentences like "I KNOW WHO ...", again based upon the state IDENTITY. Here an over-literal translation from the conceptual representation would yield "I KNOW THAT SOMEBODY IS THE PERSON WHO ...". These alterations are carried out by small - typically 15 line - program segments. Implementation of such editing processes requires little new machinery besides the provision of some function able to traverse "chains" in the syntax net, analogous to the paths in the conceptual structures.

3.4.2) Noun phrase production

Because of the complexity of the production of noun phrases, this topic was not discussed in section 3.3. The generator has a wide range of ways of expressing noun phrases, many of which are subject to quite complex constraints, but which, taken together, constitute a respectable assault on the problems of reference; and yet there are significant omissions. Specifically, the generator is currently limited to concrete nouns, though it is believed that an extension to allow abstract nouns would be quite feasible.

The program derives, as discussed above, information about the desired syntactic function of a noun phrase, and paired with this is a specification of the corresponding subpart of the conceptualisation. This specification is in the form of a path through the conceptualisation, and will point to a structure, which, in the simplest case, will have the form "(CHILD1)", as in the example above; or a more complex structure headed by FOCUS, eg

```
((FOCUS (STATE THING))
 (STATE (STATENAME QUANTIFY)
        (THING BANANAS1)
        (VAL *MANY*)
        (TIME ...)))
```

also in the above example.

In either case, the "token" is extracted, ie "CHILD1" or "BANANAS1", and is used to find all associated information which has been collected in the canonicalisation phase mentioned earlier. When this has been done, this qualifying information is removed so that multiple references to the same token do not cause repeated addition of the same qualifying information. The link between the current clause and a noun phrase being processed involves the use of three syntax nodes, which we can call "lexnoun-node", "nominal-node" and "english-node".

The english-node contains the word, or words, which correspond to the token. The nominal-node corresponds to the noun-phrase as a whole, and contains at least a link to the english-node. This nominal-node is used whenever a reference to that noun phrase is needed within the one sentence. The lexnoun-node corresponds to a particular use of this noun-phrase, and contains at least a link (LEXNOUN . <nominal-node>). Additionally it may contain reference to prepositions, eg (PREP . FROM), or other words, all specified by the verb-sense case frame. The clause-node will contain a link (<syntax-rel> . <lexnoun-node>), serving to define the function of the noun phrase in the current clause.

Thus, if a nominal-node already exists for this token, it is located and immediately bound to the newly-created lexnoun-node. Otherwise, a nominal-node and an english-node are created and endowed with suitable properties, and a scan is made for extra qualification.

Any qualifying information about the token, which has been gathered during the canonicalisation phase and placed on "global-subpropositions", is then categorised. The available categories are

i) POSSESSORS : corresponding to the conceptual pattern

```
((FOCUS (STATE THING))
 (STATE (STATENAME POSS | PART | OWN)
        (THING ...)
        (VAL   ...)))
```

ii) ATTRIBUTES : corresponding to the conceptual pattern

```
((FOCUS (STATE VAL)
 (STATE (STATENAME POSS | PART | OWN)
        (THING ...)
        (VAL   ...)
        (TRUTH FALSE) - optional extra
```

iii) ADJECTIVES : corresponding to any scale state.

iv) RELATIVES : for anything else.

These categories are then separately treated as follows.

3.4.2.1) Possessors

There are three ways in which possessors may be treated, which I shall illustrate by example.

1) FRED'S CAT

2) THE HAIR OF THE DOG THAT BIT YOU

3) THE CAT WHICH BELONGS TO FRED
THE CAT WHICH FRED HAS
THE HAIR WHICH IS PART OF THE DOG

Mode (1) can only be used when the possessor can be expressed without post-modification. Thus any information about the possessor must be either adjectival, or possessive. This entails a recursive inspection of qualifying information about the possessor (if any) of the possessor. This means that the program is able to say things like "FRED'S BROTHER'S CAT" or even "FRED'S BIG BROTHER'S CAT", etc.

Mode (2) must be chosen when the possessor must be post-modified. Expression in mode (3) is only selected when a special debugging flag, "distinguishing-withs", is set, requesting that possessors and attributes should be clearly distinguished, and is only used when it is suspected that the analyser is faulty. Possessive and attributive relationships are then expressed as relative clauses, causing different verbs to be chosen to express the relationship.

When a possessive relationship, of whatever form, is added to the syntax net, the token corresponding to the possessor is treated as a noun phrase immediately, and so will have its own lexnoun-node,nominal-node and english-node. This is true for all the types of non-adjectival modification.

3.4.2.2) Attributes

This conceptual pattern will, in the general case, promote production of syntax-net structures corresponding to the phrases "X WITH Y" or "X WITHOUT Y". This is achieved primarily by the use of two more syntax-net functions, named WITH and WITHOUT, each taking a syntax-node as argument.

There are two improvements upon this, both concerned with the conceptual PART relationship. One concerns the conceptual information "Y is not part of X", where Y is a simple, ie unqualified, nominal: this can be better expressed as "Y-LESS X". The other is the special case , "Y is part of X", where Y is itself modified by a single adjective: this is better expressed as " <adj>-Y-ed X". These transformations are effected by inspecting the syntax-net structure which has been created, and effecting appropriate modifications to this structure. One example should suffice to show both these phenomena. From the conceptual structure shown overleaf, the generator produces the sentence:
"THE BIG WINDOWLESS TRUCK BUMPED INTO THE LONG-HAIRED GIRL."

```
((EVENT (ACTOR *MOMENTUM*)
        (ACT PROPEL)
        (OBJECT (FOCUS (STATE THING))
                (STATE (STATENAME SIZE)
                       (THING (FOCUS (STATE VAL))
                              (STATE (STATENAME PART)
                                     (THING WINDOWS2)
                                     (VAL TRUCK1)
                                     (TIME (NAMED TIMEPOINT2)
                                           (COMPARISON
                                               (BEFORE *NOW*)))
                                     (TRUTH FALSE)))
                       (VAL 3)
                       (TIME (NAMED TIMEPOINT2)
                             (COMPARISON (BEFORE *NOW*)))))
        (FROM DUMMY-PLACE1)
        (TO (FOCUS (STATE VAL))
            (STATE (STATENAME PART)
                   (THING (FOCUS (STATE THING))
                          (STATE (STATENAME LENGTH)
                                 (THING HAIR1)
                                 (VAL 3)
                                 (TIME (NAMED TIMEPOINT2)
                                       (COMPARISON
                                           (BEFORE *NOW*)))))
                   (VAL GIRL1)
                   (TIME (NAMED TIMEPOINT2)
                         (COMPARISON (BEFORE *NOW*)))))
        (TIME (NAMED TIMEPOINT2)
              (COMPARISON (BEFORE *NOW*)))))
```

Map of syntax net (The top node is Syntax-node-1)

```
Syntax-node-1 contains:  ((ACTSBJ . Syntax-node-2)
                          (TENSE . PAST)
                          (FORM)
                          (LEXVERB . BUMP)
                          (OBJ . Syntax-node-8))
Syntax-node-2 contains:  ((LEXNOUN . Syntax-node-3))
Syntax-node-3 contains:  ((ADJ BIG)
                          (ADJ WINDOWLESS)
                          (ENGLISH . Syntax-node-4))
Syntax-node-4 contains:  (TRUCK)
Syntax-node-5 contains:  ((LEXNOUN . Syntax-node-6))
Syntax-node-6 contains:  ((ENGLISH . Syntax-node-7))
Syntax-node-7 contains:  (WINDOWS)
Syntax-node-8 contains:  ((PREP . INTO)
                          (LEXNOUN . Syntax-node-9))
Syntax-node-9 contains:  ((ADJ LONG-HAIRED)
                          (ENGLISH . Syntax-node-10))
Syntax-node-10 contains: (GIRL)
Syntax-node-11 contains: ((LEXNOUN . Syntax-node-12))
Syntax-node-12 contains: ((ADJ LONG)
                          (ENGLISH . Syntax-node-13))
Syntax-node-13 contains: (HAIR)
```

This example illustrates the way in which a syntax net is modified: nodes 5,6 and 7 were created but are now redundant and no longer accessible.

3.4.2.3) Adjectives

Apart from the mode of adjective formation from PART just mentioned, adjectives are derived from states other than { POSS, PART, OWN, MLOC, LOC, IDENTITY }. Such states are usually in the form of scales, ie the VAL will be in the range -10 to 10. Exceptionally, as in the case of COLOUR, the English word occurs directly in the VAL slot.

For scales, the number found is looked up in a file keyed on the STATENAME, which typically has the form

```
((SCALE ( adj1 num1 adj2 num2 adj3 num3 ...))
 (UP (adj4))
 (DOWN (adj5)))
```

In the case where the VAL is a simple number, it is compared against num1,num2 etc until a number is found which does not exceed it. The adjective which precedes that number is then selected. This parallels BABEL's method of handling adjectives.

In the case where the VAL has the form (HIGHERBY ...), the adjective indicated by UP is selected; similarly, DOWN corresponds to (LOWERBY ...). It should be noted that comparatives cannot yet be handled.

3.4.2.4) Relative clauses

Relative clauses are produced for those forms of qualification which do not fall into any of the categories listed above. For relative clauses, this qualifying conceptual information is filtered through a discrimination net, just as for the sentence and ordinary subordinate clauses, and the syntax net augmented accordingly. During this process, the fact that the relativised object is indicated by FOCUS in the sub-conceptualisation ensures that it will not be omitted from the resulting syntax net, even if it is a dummy, as a consequence of either verb-sense selection or optional syntax-roles.

The clause-node for the conceptualisation which has just been processed will contain pointers to other nodes, and will at this stage be equivalent to an ordinary sentence. This clause-node may be called the "rel-clause-node". One of the pointers from it, in the simple case to be discussed first, will denote a structure of a newly-created lexnoun-node, the original nominal-node and the original english-node. To avoid confusion, let the newly-created lexnoun-node be called "rel-lexnoun-node".

The situation may be envisaged thus:

```
  top-clause-node:  (...
                     (syntax-rel . lexnoun-node)
                      ...)
     lexnoun-node:  ((PREP . TO)    <- optional
                     (LEXNOUN . nominal-node))
     nominal-node:  ((ADJ . BIG)    <- optional
                     (ENGLISH . english-node))
     english-node:  (BANANAS)
  rel-clause-node:  (...
                     (syntax-rel . rel-lexnoun-node)
                      ...)
 rel-lexnoun-node:  ((LEXNOUN . nominal-node))
```

For simple relative clauses, the operations that must be done are threefold:
-Associate rel-clause-node with the token
-Reorder rel-clause-node so that the reference to rel-lexnoun-node comes first (since rel-clause-node has already been sorted at this stage)
-Doctor rel-lexnoun-node so that a relative pronoun is used.
For simple cases, the procedure is:

i) Add the pair "(REL . rel-clause-node)" to the end of nominal-node.

ii) Reposition the pair "(syntax-rel . rel-lexnoun-node)" at the top of rel-clause-node

iii) Change LEXNOUN to WHINDIC on rel-lexnoun-node. WHINDIC is a function which produces the relative pronouns WHO, WHOM, WHICH and WHERE , making its choice on the basis of the syntax-relation and the properties of the token.

Thus, when the syntax net is "evaluated", the English name of the token is followed by the entire relative clause, which may, depending on syntactic context, be enclosed in commas. The fact that prepositions are attached to lexnoun-nodes ensures that, if the relativised object is dominated by a preposition in the relative clause, the preposition is correctly placed after the relative pronoun.

There are three refinements to this procedure for less simple cases. Firstly, some case-frames specify prepositions which should be dropped if the object is relativised. For instance, compare "I WENT TO PARIS" and "PARIS, WHERE I WENT" This is handled by having a syntax-relation NOTFRONTPREP, which is stripped off rel-lexnoun-node if it is present.

Secondly, there are instances where multiple relative clauses are needed. For the second and subsequent relative clauses, the relation ANDREL is placed at the end of nominal-node. The actual mechanism for this involves always using ANDREL, then changing the first to REL at the end of the process.

The third refinement is more complex. Some relative clauses may contain their only reference to the relativised object inside a subordinate clause. Such cases, which Winograd refers to as DOWNREL, appear to be confined to situations such as

THE MAN WHO, FRED TOLD MARY, HAD GONE TO THE PARK.

which is derived from

FRED TOLD MARY THAT THE MAN HAD GONE TO THE PARK.

The straightforward subordinate relationship is handled by the syntax relation "S2", whose effect is to prefix the embedded clause with the word "THAT". In such a case, the reference to rel-lexnoun-node will not be found on rel-clause-node; but rather will be found on the node indicated, on rel-lexnoun-node, by S2. To handle this problem, the program will locate rel-lexnoun-node on this node, and in addition to performing all the normal actions will alter S2 to S3. Since this involves stripping a pair from the subordinate-clause node and placing it at the head of rel-clause-node, occasionally, as in the example cited, the subject of that clause may be removed. The effect of this will be that at net "execution" time, the number is unknown (since ACTSBJ has the side-effect of discovering and communicating number). So, if this occurs, ACTSBJ is not deleted, but is replaced with a special function TRACE-ACTSBJ which has the same properties.

As a final direct illustration of the generator, a simple sentence with a relative clause would be produced from an input conceptualisation as shown overleaf:

Conceptualisation:

```
((STATE (STATENAME ANGER)
        (THING (FOCUS (CAUSE RESULT EVENT TO))
               (CAUSE
                   (ANTECEDENT
                       (EVENT (ACTOR FRED)
                              (ACT PTRANS)
                              (OBJECT (FOCUS (STATE THING))
                                      (STATE (STATENAME PART)
                                             (THING HANDY-HAND1)
                                             (VAL FRED)
                                             (TIME (NAMED TIMEPOINT1)
                                                   (COMPARISON
                                                      (BEFORE *NOW*)))))
                              (FROM FRED)
                              (TO MAN1)
                              (TIME (NAMED TIMEPOINT1)
                                    (COMPARISON
                                       (BEFORE *NOW*)))))
                   (RESULT
                       (EVENT (ACTOR *MOMENTUM*)
                              (ACT PROPEL)
                              (OBJECT HANDY-HAND1)
                              (FROM FRED)
                              (TO MAN1)
                              (TIME (NAMED TIMEPOINT1)
                                    (COMPARISON
                                       (BEFORE *NOW*)))))))
        (VAL 3)
        (TIME (NAMED TIMESPAN1)
              (TS (NAMED TIMEPOINT8)
                  (COMPARISON
                     (BEFORE *NOW*))))))
```

The first step is to canonicalise the conceptual structure, collecting together all the time references, and putting the entire substructure headed by FOCUS onto the list "global-subpropositions". Then, an appropriate discrimination net is sought for the main conceptual structure; the one initially selected is the one used for all states.

This discrimination net first discovers that the time reference is to a TIMESPAN with a definite starting time, but an indefinite end. It checks against a few special cases - for instance, had the STATENAME been POSS, the verb ACQUIRE would have been selected - but finds that none of these apply; consequently, it chooses the verb sense BECOME1, whose case frame is:

```
(BECOME1                          ; the lookup-key
   BECOME                         ; the verb stem
   ((ACTSBJ (STATE THING))        ; syntax role ACTSBJ, and path
    (P_ADJ NOTHING (ONSCALE)))    ; an unusual syntax role, see below
   (TIME-SOURCE (STATE TIME TS))) ; A "special action", see below
```

(This case frame has two special features. First, no path is specified for the P_ADJ role, but the function ONSCALE is to be applied; second, a "special action" is given, indicating the time to be used for tensing purposes, and overriding the general-purpose defaults.)

A new syntax node, Syntax-node-1, is created; the first activity is to add to it the role-value pairs

```
(LEXVERB . BECOME)     ; the verb
(TENSE . PAST)         ; because of the relationship between
(FORM . NIL)           ;  TIMEPOINT8 and *NOW*
```

Then the specific contents of the case-frame are processed; ACTSBJ is one of the roles which produce noun phrases, and so the nodes Syntax-node-2, Syntax-node-3 and Syntax-node-4 are created: in the terminology introduced in section 3.4.2, these are the lexnoun-node, the nominal-node and the english-node respectively.

The path specification (STATE THING) is followed, and leads to MAN1. This contains a (hidden) pointer to the token MAN, for which the appropriate English word is MAN; so the english-node is set to hold the symbols (MAN). (This is a list, because one could have, say, LAMP POST) The nominal-node is then set to contain a link to the english-node, and to any other information known about the token MAN1. Such information is known, and was placed on "global-subpropositions" during canonicalisation. This information is retrieved - and deleted from that list - and is inspected to see what sort of information it encodes. Since the modifying structure is not a state at all, it cannot be possessive, attributive or adjectival in nature; so a relative clause must be formed.

The formation of a relative clause involves first the production of a clause: discrimination net application, case frame retrieval, and syntax net construction; all of which proceed without significant diversion in this example. When this process is completed, the syntax node "representing" the clause, Syntax-node-5, is sorted so that the syntactic labels are in the standard order. Then the transformation of this, simple, clause into a relative clause begins.

At this point, the syntax net is as shown below:

```
Syntax-node-1 contains:  ((TENSE . PAST)
                          (FORM)
                          (LEXVERB . BECOME))
Syntax-node-2 contains:  NIL
Syntax-node-3 contains:  ((ENGLISH . Syntax-node-4))
Syntax-node-4 contains:  (MAN)
Syntax-node-5 contains:  ((ACTSBJ . Syntax-node-6)
                          (TENSE . PAST)
                          (FORM . PROG)
                          (LEXVERB . PUNCH)
                          (OBJ . Syntax-node-9))
Syntax-node-6 contains:  ((LEXNOUN . Syntax-node-7))
Syntax-node-7 contains:  ((ENGLISH . Syntax-node-8))
Syntax-node-8 contains:  (FRED)
Syntax-node-9 contains:  ((LEXNOUN . Syntax-node-3))
```

Firstly, the english-node is given the extra pair (REL . Syntax-node-5). The nominal-node, Syntax-node-3, is then compared with all the nominal-nodes accessible from Syntax-node-5, permitting just access via a lexnoun-node. The pair (OBJ . Syntax-node-9) is found to indicate the same nominal-node, and so indicates the link between the relative clause and the relativised object. The rel-clause-node, node 5, is then reordered with this pair at the front; this is step 2. Finally, Syntax-node-9 is modified so that, rather than saying (LEXNOUN . Syntax-node-3) it says (WHINDIC . Syntax-node-3); this, the final step in simple relative-clause processing, ensures the use of a relative pronoun.

The simple noun-phrase procedures are now resumed: the lexnoun-node is given a link to the nominal-node, and any prepositions (here there are none) are also added to the lexnoun-node; it is sorted; and then the clause node - Syntax-node-1 - is linked to it via the appropriate syntax relation.

The other element of the case-frame for BECOME, P_ADJ, is much simpler, but unusual. It specifies that no part of the conceptual structure should be used to fill this slot; but also specifies that the function ONSCALE is to be applied. ONSCALE is the function which retrieves adjectives, and here selects the adjective FURIOUS. When this has been done, the top-level syntax node is itself sorted, and the final structure of the syntax net is as reproduced on the next page.

```
Syntax-node-1 contains:  ((ACTSBJ . Syntax-node-2)
                          (TENSE . PAST)
                          (FORM)
                          (LEXVERB . BECOME)
                          (P_ADJ . Syntax-node-10))
Syntax-node-2 contains:  ((LEXNOUN . Syntax-node-3))
Syntax-node-3 contains:  ((ENGLISH . Syntax-node-4)
                          (REL . Syntax-node-5))
Syntax-node-4 contains:  (MAN)
Syntax-node-5 contains:  ((OBJ . Syntax-node-9)
                          (ACTSBJ . Syntax-node-6)
                          (TENSE . PAST)
                          (FORM . PROG)
                          (LEXVERB . PUNCH))
Syntax-node-6 contains:  ((LEXNOUN . Syntax-node-7))
Syntax-node-7 contains:  ((ENGLISH . Syntax-node-8))
Syntax-node-8 contains:  (FRED)
Syntax-node-9 contains:  ((WHINDIC . Syntax-node-3))
Syntax-node-10 contains: ((ADJ FURIOUS))
```

Executing this "program" produces the sentence
THE MAN,WHOM FRED WAS PUNCHING,BECAME FURIOUS.
Further examples of the output of the generator are given in chapter 6, where the performance of the system as a whole is illustrated.

3.5) Scope for further work

The techniques used in BABEL have been largely carried over into AD-HAC's generator, with some modification and resultant improvement. The present generator tackles some problems which BABEL did not, but has left several others for future attention. This section identifies some of these problems, pointing out where possible how the existing program would have to be modified in order to solve them.

3.5.1) Abstract nouns

In English, the verb of a sentence usually carries the major responsibility for conveying meaning, while nouns can be slotted in to appropriate syntactic positions; however, it is easy to find sentences where it is not the verb which conveys meaning, but one of the nouns: in particular, there are almost meaningless verbs which positively require this. For example,

THE SOVIET INVASION OF CZECHOSLOVAKIA OCCURRED IN 1968.

In this sentence, the main verb - OCCURRED - says very little; most of the meaning is conveyed by the noun INVASION, which is closely related to the verb INVADE. Such nouns are abstract nouns, rather than concrete nouns.

Abstract nouns can be used to shift the emphasis in a sentence, and their use makes texts correspondingly more fluent. The current generator is unable to use abstract nouns, though I believe the modifications required would be quite simple (though extensive). Briefly, abstract nouns are very similar to verbs: they convey meaning in the same way as do verbs, and they may be associated with case frames in the same way as verbs. The principal difference, from the point of view of a generation program, seems to be that the selection of a verb as the main information carrier in a sentence guarantees that a syntactically well-formed sentence can be produced; while selecting an abstract noun still leaves open the problem of selecting a verb, even though it will often be a neutral one like "occur".

Since abstract nouns are so similar to verbs, and indeed are usually closely related to verbs, they could be incorporated into the present program as follows. In the discrimination nets which select verb senses, tests could be made to see if some higher-level clause was actively seeking a noun rather than a verb; and if so, returning a case-frame pointer for the associated abstract noun rather than for the verb. In the program as it stands, there are already facilities whereby tests in the discrimination nets may involve seeing what answer another discrimination net will deliver; this could easily be extended so that some declaration of what was wanted could guide the selection in the lower network.

This scheme would involve the provision of case frames for abstract nouns, and would allow higher-level verb-selection decisions to depend on the availability or otherwise of abstract nouns to express some embedded conceptualisation. Two points are worth noticing about this proposal: firstly, it would permit the use of abstract nouns to replace embedded clauses, but would not account for the use of abstract nouns as the major meaning element of top-level clauses; secondly, it seems likely that the syntax roles used in the case frames would have to be different from those in use for verbs.

3.5.2) Translation into other languages

The generator described here produces only English. It forms part of a larger system which understands stories stated in English, and is typically used by the inference mechanism for such tasks as expressing the answers to questions, reporting the identification of pronoun referents, and in the most general case, expressing arbitrary inferences. As noted in the introductory chapter, it is possible to link the generator directly to the sentence analyser, thus getting a line-by-line paraphrase of a story: it would be trivial to have the generator paraphrase the story on a line-by-line basis after the inference mechanism had done its work, and thereby get the benefits of the programs pronoun-resolution capability reflected in the paraphrases.

An interesting extension of this idea is to have the generator produce some different language, rather than the language in which the story was told, and thereby producing translations of the stories: many would regard this as a stronger demonstration of understanding than merely, as at present, answering questions and such like.

AD-HAC's generator has in fact been modified to produce Japanese [Abe, 1980], and it is my belief that it could be modified, with little further effort, to produce sentences in other languages too.

3.5.3) <u>The interface with the inference mechanism</u>

As Goldman pointed out, good generation requires the exploitation of the inference mechanism, and particularly of its associated memory; only by these means, for instance, can the generator select the verb RETURN rather than GIVE to express a particular act of giving.

In AD-HAC, the interface between the generator and the inference mechanism is extremely crude. The facility, suggested by Goldman, of permitting the generator to ask for something matching a specified conceptual pattern, has been implemented in a rudimentary fashion; but since the generator is often used to express inferences "on the fly", the existing implementation often gets in the way of the inferencers normal work: consequently, the facility is in practice little used.

3.5.4) <u>Relaxing the sentence-per-CDform constraint</u>

AD-HAC's generator, like BABEL, takes one CDform as its input, and produces one sentence as its output. This simplifies the generation task enormously, yet seems hardly realistic. One consequence of this approach is that most of the sentences produced by the generator are extremely short, because it is given only one simple thing to say; but some sentences are ridiculously long because the generator has been given a complex CDform to express, and many of the nominals it will use need to have modifiers attached to them.

The roots of this problem lie in the fact that the generator does not have any metric for judging the complexity of the sentences it produces. If it had, and if it could be guaranteed that the successive sentences it was going to produce belonged together in a coherent text, then the generator could simply collect sentences together until some threshold of complexity was reached, and then conjoin all these simple sentences to make one larger one; in the case where it was called on to express some CDform which was too complex, it could split the unwieldy sentence into parts.

However, such a metric is not easy to find. The first approximation, a simple word count, is inadequate, as inspection of any realistic text will show.

3.5.5) Pronominal reference to sentential antecedents

The current generator's approach to pronoun use is extremely simple. Pronouns are produced when a nominal token is used for the third and subsequent times in a sentence, or when the referent of the token needs to be expressed as "I" or "YOU".

This approach is much too simple, even for the simplified case of reference to nominal tokens. More significantly, it completely rules out the use of pronouns to refer to previously related events. Extending the generator to use pronouns with sentential antecedents would involve both equipping the generator with a memory for what it had already said, and dealing with many of the problems which would have to be tackled in using abstract nouns.

Chapter 4 - Analysis 1: Background

A great deal of effort has been directed at the analysis of natural language, mostly concerned with English, and a correspondingly large number of analysis programs have been written. As time has passed, attention has shifted from purely syntactic parsing to semantic analysis.

The very profusion of analysis programs precludes a discussion of all of them here. The first section of this chapter gives a very brief historical survey, whilst the second section describes more fully those aspects of particular analysers which influenced the design of that used in AD-HAC. This entire chapter is merely a prelude to the next, which describes AD-HAC's analyser in detail.

4.1) Some early approaches to analysis.

This section discusses the analysers built by Kuno and Oettinger, by Kay, and by Winograd. I have selected these because, in their own terms and in their own times, they were highly influential. Many of their innovations have had consequences which may be traced through to the present, but which are hard to acknowledge properly because they have become assimilated into the folklore.

Because none of the analysers discussed in this section have had any direct bearing on the design or implementation of AD-HAC, the discussions here are brief to an extreme: extensive descriptions may be found in [Kuno and Oettinger, 1962], [Kay, 1964], [Kay, 1973], [Winograd, 1971].

4.1.1) Kuno and Oettinger: The Harvard Predictive Analyzer

One of the earlier analysers was the Harvard one of Kuno and Oettinger, which was designed for machine translation. It was focussed on syntax, and was based on a technique of predictive syntactic analysis. Prior to the development of this analyser, the application of this technique had been frustrated by an inability to discover more than one analysis for a given sentence; part of the significance of this analyser lies in its application of parallel processing techniques to sentence analysis.

For this analyser, as for its progenitors, a "grammar" consisted of a set of production rules, and a set of syntactic labels was associated with words in the language. Initially, productions which predicted that label found on the first word of the sentence were placed in the "prediction pool", and constituted the "grammar" for the remainder of the sentence. (The entire grammar could be regarded as forming the initial prediction pool.)

Kuno and Oettinger introduced a number of refinements to this basic technique, facilitating the production of multiple (alternative) analyses. Firstly, they transformed the grammar from a set of productions into an array, keyed on structures predicted and syntactic labels found, and containing in each element of the array a set of predictions, each of which was a sequence of expected structures. Secondly, they split the prediction pool into a number of subpools, each corresponding to a prediction. These subpools may generate new subpools if their topmost element is compatible with the next word, or be discarded otherwise.

4.1.2) Kay: The Chart Parser

Still within the syntactic analysis paradigm, Kay introduced the algorithm which he called a "Chart Parser". This built upon an earlier development, the "well-formed substring table", which eliminated the need to process the same string of words more than once when trying alternative predictions. The chart parser algorithm builds a structure which initially contains, associated with each word, a set of primitive syntactic labels. These may be augmented with higher-level syntactic labels which may correspond to more than one word, such as NP, PP etc., which may themselves generate the sentence symbol, S. By keeping track of the derivation of higher-level symbols, this recognition capability is easily extended to a parsing capability, ie. the ability to generate syntactic structures representing the sentence.

4.1.3) Winograd: SHRDLU

Whereas previously it had generally been assumed that semantic procedures should be used to select between competing syntactic readings of a whole sentence, Winograd utilised semantic judgement to guide syntax-based analysis, eliminating the production of many implausible syntactic readings. He used Halliday's "Systemic Grammar", which emphasises characteristics of clauses, phrases or sentences. His parsing routines were written in a language called PROGRAMMAR which was tailored to the operations needed for conducting a systemic analysis. This style of analysis is heavily top-down in its general nature, since the "systems" specify sets of alternative features, but Winograd was concerned to rely upon intelligent backtracking rather than blind search; and so he implemented a system of "messages" which could communicate not only that a given feature was not present, but could be used to determine which feature most probably was present.

4.1.4) Summary

Although these earlier analysers were dissimilar on a gross plane, it is apparent that there were common threads running through: for instance, the notion of prediction, and the handling of multiple syntactic readings. These shared features contribute greatly to the background of folklore on which later work must build. The analysers described in the next section, those of Riesbeck, Wilks and Woods, draw on this, and themselves form the background to my own work which is described in the next chapter.

4.2) Analyers particularly relevant to AD-HAC

This section discusses in some detail the analysers constructed by Riesbeck, Wilks and Woods. These three have had significant influence on the construction of the analyser used in AD-HAC, each having some features considered desirable but others considered less so. Naturally, it has been a goal of mine to combine the strengths whilst eliminating the weaknesses. The next chapter summarises, in the form of a set of guiding principles, the lessons learnt from these particular programs.

4.2.1) Riesbeck

Riesbeck constructed an analyser [Riesbeck 1974] which was intended to be a model of human language understanding, producing a purely semantic representation of the sentences it parsed in a single left-to-right pass. Specifically he hoped to find the meaning of a sentence without having to sift through syntactic alternatives at any stage.

In achieving these goals, he developed the notion of "expectation", which was already present, though in a purely syntactic form, in the work of Kuno and Oettinger (see above). In Riesbeck's work, syntactic prediction was replaced by semantic expectation. Whereas earlier the predictions were expressed as a general grammar, Riesbeck's expectations were more closely associated with the individual words in the dictionary. Naturally, some of the semantic expectations were rather vacuous, such as those attached to determiners; but for verbs especially, they could be quite specific.

Expectation may provide a solution to determining the intended senses of subsequent words, but a parser must also build some structure, be it semantic or syntactic, to represent a sentence; for otherwise it can, at best, only recognise a string as meaningful/meaningless or well-/ill-formed. The "requests" in Riesbeck's system were therefore a combination of expectations and actions, and were associated with the words in the dictionary. These requests had two parts, a test, corresponding to the expectation, and an action, a piece of program which was executed when the expectation was satisfied. These actions were unrestricted, and in particular could add new requests to the current list, as well as building and modifying semantic structures expressed in Conceptual Dependency.

The basic operation of Riesbeck's program, simplified to consider only declarative sentences, was as follows. Suppose the sentence was "JOHN GAVE MARY A BOOK". Initially, there would be a request whose "test" portion corresponded to an expectation for a noun phrase, and whose "action" would store this noun phrase somewhere and added a new request looking for a verb. For the example sentence, 'JOHN' will somehow satisfy the request for a noun phrase, and so the request will be used and then deleted. Additionally, any requests attached to 'JOHN' will be added to a list of requests, but I shall assume here that there are none. Then the next word, 'GAVE', is inspected, and is found to satisfy the request for a verb; so the action part of this request is executed.

At this point, the word 'GAVE' has been accepted, and the request list is empty. The requests associated with 'GAVE' are loaded. One of these has a test which is immediately satisfied without inspecting more words, and produces a semantic structure corresponding to giving things to people, also moving the structure for 'JOHN' from its temporary location into the correct place - the ACTOR and FROM slots - in this structure. Another request will be looking for a further noun phrase with the feature "HUMAN" for the recipient, and yet another will look for a noun phrase with the feature "PHYSICAL OBJECT" for the object (*1).

Following through this example, the noun phrases 'MARY' and 'A BOOK' activate requests which are specific to the verb GIVE, whose associated actions augment the semantic structure built by 'GAVE' with appropriate

---

*1 Other requests, not used in this example, would be used to pick a particular sense of the word 'TO', or to handle sentences like JOHN GAVE MARY A BEATING "picture producers" - these being the structures used to represent concrete nouns in CD. Finally, the full stop is accepted and causes the analysis to terminate. Throughout this processing, the verb assumes a central role: both in terms of the construction of a semantic structure, and in terms of providing the information which guides the analysis of the remainder of the sentence.

The outline given above, though very brief, serves both to introduce some terminology and to provide a background for a critical discussion. There are several features of this system which I consider desirable, but also some which I believe should be eliminated. Desirable features are

+i) The flexibility of the request mechanism.

The association of unrestricted actions with arbitrary tests will enable a program to respond appropriately in very complex situations.

+ii) The direct transition from surface form to meaning representation.

For the normal purposes of communication, conventional syntactic structures are of little interest, except insofar as they may give a clue to the attitudes of the speaker/writer; and even this function may be performed by suitable requests. If it is possible to analyse sentences directly for content, exploiting syntactic clues but without first generating intermediate syntactic structures, this appears to be the correct way to proceed.

+iii) The central role of the verb.

Though the situation is complicated for abstract nouns, in mundane sentences involving physical objects the principal meaning content is often determined by the verb, the primary role played by the objects being to select between alternative senses of the verb or alternative case-frames. It seems sensible to embody principles such as these in the operation of a language analyser.

Features considered undesirable are

-i) Purely deterministic operation.

One of the goals of Riesbeck's work was to provide a computational model of human language understanding, and he took this to mean that a single left-to-right scan was mandatory. Although he accepted that multiple readings are occasionally valid, and that it is not always possible to select the correct interpretation first time, he did not allow for any form of backtracking at all. Essentially, his approach in situations where nondeterminism would be useful was to provide a specification of how to correct, on the spot, an analysis which was incorrect. And this despite his protestations that

> "The use of ... a decision point mechanism would be making an implicit claim, that when people make decisions, they expect them to go wrong." (p.41).

-ii) Awkwardness of word-by-word processing.

The absence of any standard syntactic recognition component means that strings of words cannot easily function as units. Each determiner, for instance, must carry requests which invoke noun-phrase finding routines. Similarly, the analysis of complex verb groups via the request mechanism must be exceedingly cumbersome.

-iii) Inability to remove requests except by executing them.
There was no mechanism for removing requests, unless they "fired". If a request was liable to cause problems if left around too long, the only way it could be removed was by arranging that it should "fire" in all circumstances, but only perform some real action in some of those circumstances. This led to the "action" parts of some requests in fact containing the operative predicate; which confounds the advertised standard predicate-action pattern.

-iv) Fiddling with word definitions.
Though not illustrated in the example above, one of the requests loaded by 'GAVE' can set the preferred interpretation of the word 'TO'. It does this by rearranging the dictionary definition of the word; this has then to be reset in some way before the analysis of the next sentence. (It is not clear how the analyser would cope with a sentence where the word 'TO' occurred twice, once in the context of 'GIVE' and once elsewhere: for instance, "JOHN GAVE THE BOOK TO MARY TO READ".) Such operations are a grave setback to the ideal of extensibility of any system, and introduce an unwarranted degree of complexity into the otherwise modular behaviour of the dictionary entries.

This work of Riesbeck greatly influenced the development of AD-HAC's analyser, which also uses requests associated with words, but overcomes all the problems mentioned. Chapter 5 describes how.

4.2.2) Wilks

Wilks produced an analyser for an English-French translation system [Wilks, 1973], [Wilks, 1975b], whose generation component has already been discussed in Chapter 3. Underlying this whole project was the notion of "preference semantics"; its application extends beyond the boundaries of analysis as narrowly conceived, and into the domain of pronoun resolution. One of the major problems in understanding language is the problem of choice: words, phrases and entire texts may be capable of more than one meaning, and any absolute rules for selecting one interpretation rather than another are remarkably elusive, if they exist at all. A natural text will however contain many clues to the intended interpretation; though any individual clue may be misleading, their cumulative effect is to prefer one interpretation over all others. This is the principle underlying preference semantics.

In Wilks's system, the representation of sentences, and indeed of whole texts, was semantic rather than syntactic in nature. An inventory of about 80 primitives was used to encode lexical information, and also to specify structural information, via deep semantic cases. The analysis of text, involving lexical and structural ambiguity, was done with primitive matching within the framework of patterns specifying collocational requirements. The system strove to satisfy these.

Many of the primitives described objects, actions and properties. For convenience in characterising patterns, some of these primitives were grouped into classes: for instance '*ANI' could refer to any of {MAN FOLK BEAST}. This was not, however, a hierarchical organisation. These primitives were combined in "Formulas" which characterised each sense of each word distinguished by the system: the formulas had a rigid internal syntax which, in particular, designated one of these primitives as the "head" of the formula. This head essentially indicated the defining characteristic of the real-world entity to which the formula referred.

Some examples of simple word-sense formulas are reproduced here: note that the head elements are at the right.

```
Interrogate: ((MAN SUBJ) ((MAN OBJE) (TELL FORCE)))
Policeman:   ((FOLK SOUR) ((((NOTGOOD MAN) OBJE) PICK) (SUBJ MAN)))
Crook (1):   ((((NOTGOOD ACT) OBJE) DO) (SUBJ MAN)))
Crook (2):   ((((((THIS BEAST) OBJE) FORCE)(SUBJ MAN)) POSS) (LINE THING))
```

The system made use of a list of "bare templates", which were triples usually of the form Actor-Action-Object. Bare templates, defining basic propositional forms as sequences of formula heads, were useful for selecting possible meanings of ambiguous words on the basis of immediate context. For instance, some bare templates are:

```
(MAN FORCE MAN)
(MAN FORCE THING)
```

For simple clauses, the structures built in the course of analysis would also consist of triples: the formulas for individual senses of words would only be combined into such a triple if their heads matched one of the bare templates. These triples of full formulas, matching some bare template, were called "templates", and could themselves be linked together by means of higher structures called "paraplates", indicating other case relationships than that encoded by the Actor-Action-Object template structure. The relationships indicated by prepositions, for instance, were coded in this way, the case primitive heading the preposition formula being the link between a pair of templates.

In order to parse a sentence to yield structures like these, it is necessary to somehow locate those words which correspond to the Actor, Action and Object (in the simplest case). To handle this task, Wilks made use of a "fragmentation routine" which broke the string of words at various points, selected largely by using key words like prepositions and conjunctions. Then the important words, verbs and head nouns, would yield up a number of formulas which encoded their various senses. The heads of these formulas were compared against the stock of bare templates, and sequences of formulas would be built into triples if some appropriate bare template was found. Similar operations would be performed within the individual fragments, thus associating senses of modifiers with senses of nouns for example.

At this point, it can readily be imagined that complex sentences would give a seething mass of competing templates, and that the remaining task is to select those which correspond to the intended meaning. The bare templates do however ensure that many absurd interpretations are never even considered. The collection of instantiated templates is now scanned with the goal of detecting further agreements between what is found and what is expected: An example given by Wilks,

"The policeman interrogated the crook",

demonstrates the principle of this operation. The formulas for the word senses are as given above, and two bare templates, (MAN FORCE MAN) and (MAN FORCE THING) will have been matched and then instantiated by substituting the corresponding word-sense formulae. Now, the two senses of "CROOK" have the heads 'MAN' and 'THING'; "POLICEMAN" has only one sense, head 'MAN'; and "INTERROGATE" has the single sense, headed by 'FORCE', whose formula is

((MAN SUBJ) ((MAN OBJE) (TELL FORCE))).

In this formula, 'SUBJ' and 'OBJE' are case relation primitives. The pair (MAN SUBJ) indicates that the preferred subject will have head 'MAN'; both instantiated templates have a subject whose head is indeed 'MAN'. Similarly, (MAN OBJE) indicates a preference for an object having head 'MAN'; but now only one of the instantiated templates has an object which satisfies this preference. As indicated, the formula for "INTERROGATE" expresses a preference for, rather than a restriction of, certain features of the surrounding words. Since one of the templates satisfies this, that template is "preferred"; in this example, this is the only preference which discriminates at all, and so it determines the correct interpretation of the word "CROOK". More complex sentences would be handled by counting the number of preferences satisfied, selecting that sequence of templates (and formulas) which satisfied most; applying paraplates will clearly have similar effects, leading to a gradual sorting out of the possible sentence interpretations.

This is one aspect of the process of exploiting redundancy, where redundancy takes the form of predicating descriptors which already exist: in this example, the element (MAN OBJE) is predicating the descriptor 'MAN', which is found as expected. The same principles guide all the operations performed by the analyser, including pronoun resolution, which may involve a variety of pattern-matching procedures, and may even include the application of "common sense inference rules".

To summarise the lessons drawn from studying this analyser, the most valuable feature is:

+i) The notion of preference.

The application of preference, as opposed to selectional restrictions, combines the ability to discriminate senses of words according to contextual cues with an extraordinary resilience in the face of violation of those preferences. Though the precise set of semantic primitives used may be changed, and though the organisation of these primitives may be reconfigured, preference as a guiding principle of analysis will retain its value.

There are however several aspects of this system which appear unsatisfactory, and in the construction of the analyser for AD-HAC I was determined to avoid these. They are:

-i) The fragmentation process.

This appears to be based upon a collection of ad-hoc rules, rather than being motivated by any theoretical considerations.

-ii) Processing by triples.

Assuming that Actor-Action-Object is the fundamental structure, which can fit all circumstances, perhaps by supplying dummies, is clearly mistaken, and furthermore there is no magically correct number of elements which will work for all cases. This is substantially a remark about the semantic representation employed in Wilks's project; but the triple has been allowed to dominate the design of the analyser also. The result of this is an artificial treatment of many phenomena.

-iii) The multiple "cycles of mapping".

This system was not intended to be a model of model human language comprehension, but to be a working translation program. Whereas I earlier criticised Riesbeck's rigidity on left-to-right processing, I am equally critical of Wilks's complete abandonment of any pretence of psychological plausibility.

4.2.3) <u>Woods</u>

Woods has described the LUNAR system, a natural language front-end to a database which stores information about moon rocks [Woods, 1972]. This system is radically different to those of Wilks and Riesbeck, principally in its subordination of semantic processing to conventional syntactic analysis. In this system, the role of semantic analysis was principally to validate, or to reject, the syntactic structures that are found for a sentence; and if a structure is semantically acceptable, to construct a query which is interpreted by a retrieval mechanism which inspects the data base. In Woods's system, the syntactic analyses are "deep structures" of the sort postulated by transformational grammar (TG) [Chomsky, 1965], and are generated by an ATN (augmented transition network) [Woods, 1970].

Woods asserts that, in the LUNAR context, it is computationally disadvantageous to perform semantic processing in parallel with the syntactic recognition process; disadvantageous, because slower. Consequently, it becomes important to select the correct parse first, so that the semantic routines do not have to vet large numbers of implausible readings; and this in turn demands a careful ordering of the arcs leaving particular states in the grammar. Even apart from this, there are good reasons, in the context of a sentence grammar, to order some arcs carefully: though backtracking will permit the discovery of a parse, this may be inefficient. For instance, some forms of relative clause appear superficially identical to main clauses, but are distinguished only by the presence of another main clause: so it makes sense to try to get the main clause first, since this is a mandatory component of the sentence.

It is unlikely that there exists any static ordering of arcs leaving a state which will always choose the most likely analysis first, and the ATN formalism does not allow for dynamic ordering to be applied - nor should it, for in the abstract nondeterministic machine, all arcs are of equal status. The effect of modifying the order could be achieved by providing several states with the same arcs, but in different orders, and selecting between these on whatever criteria would be used if a reordering operation were available: however I feel that the ungainliness of this solution reflects an unnatural dependence upon arc ordering.

There are many aspects of the LUNAR system which I shall not describe here because they did not influence the design of AD-HAC's analyser. In particular, the procedural semantics employed by LUNAR is not described here.

To summarise:

+i) Convenience of expression of syntactic regularities.
Where syntactic regularities are commonplace, it is highly desirable to express syntax rules in such a way that these regularities are captured: the ATN formalism does this very effectively. For instance, it provides a convenient and elegant mechanism for "parsing" verb groups and simple noun phrases.

-i) The presupposed uniformity of syntactic structures
Whilst there are undoubtedly regularities in syntactic structures, for instance, the association of objects and sometimes indirect objects with certain verbs, there are also many special cases: if a grammar is to handle these, it must both know about all the special cases, and know when they are applicable.

-ii) Special cases need new verb features.
The grammar specifies, albeit implicitly, a set of syntactic paradigms: the dictionary must supply information to enable the grammar to attempt to parse only those paradigms which are valid for a particular verb, and does this by associating "features" with these verbs. One effect of this is that each special case must be assigned a unique feature; unfortunately, though the number of special cases is very large, the grammar must be prepared to deal with each of them - there is no mechanism permitting the dictionary to supplant the grammar in dealing with special cases.

-iii) Necessity of getting all syntactic structures.
Since the ATN makes no provision for semantic judgement, there are many cases where local ambiguity cannot be resolved; for instance in the placement of prepositional phrase modifiers. For a system which is interested in the content of a sentence, such information must be accurately represented; and since there is no syntactic justification for preferring one analysis over another, in the case of multiple modifying prepositional phrases for instance, it becomes necessary to return all possible combinations for inspection by some later semantic component.

4.3) Principles guiding the design of AD-HAC's analyser

The next chapter describes the construction of a sentence analyser whose design has been heavily influenced by the considerations indicated above. This analyser, which is used in AD-HAC, attempts to draw together the merits of the analysers described above, whilst avoiding their weak points; and, as will be seen, provides some new insights into the nature of the problems of natural language analysis.

The specific influences on AD-HAC may be summarised in the form of a set of principles, drawn from considering the good and the bad aspects of the forerunners described in section 4.2. The five principles are as shown below, and carry an indication of whether the previous systems were satisfactory or not in this regard.

1) The information to guide analysis of a sentence should be derived from words in the sentence, and packaged in the form of test+action structures which we may call requests.
   (+Riesbeck)
2) However, syntactic processing should be used in a supporting role to locate groups of words which may be treated as a whole: neither a request mechanism, nor any keyword-based algorithm, should be expected to perform this task. Nor should a syntactic analysis of whole sentences be sought.
   (-Riesbeck, -Wilks, +Woods and -Woods)
3) The requests used should be manipulable and clean: capable of being removed when no longer applicable, and not modifying the definitions of other words.
   (-Riesbeck)
4) The analysers progress through a sentence should be predominantly left-to-right; yet it should be able to backtrack if necessary.
   (+Riesbeck and -Riesbeck, -Wilks)
5) There should be a mechanism for specifying and applying preferences, in relation to diverse linguistic phenomena.
   (+Wilks)

How these principles are applied to the design of a new analyser, the structure of such an analyser, and the extra details which need to be filled in, are the topic of the next chapter.

Chapter 5 - Analysis 2: AD-HAC's analyser

5.1) Principles

The previous chapter studied in some detail the analysers of Riesbeck, Wilks and Woods, and concluded by extracting five principles which should guide the design of a new semantic analyser. These principles are:

1) The information to guide analysis of a sentence should be derived from words in the sentence, and packaged in the form of test+action structures which we may call requests.
2) However, syntactic processing should be used in a supporting role to locate groups of words which may be treated as a whole: neither a request mechanism, nor any keyword-based algorithm, should be expected to perform this task. Nor should a syntactic analysis of whole sentences be sought.
3) The requests used should be manipulable and clean: capable of being removed when no longer applicable, and not modifying the definitions of other words.
4) The analysers progress through a sentence should be predominantly left-to-right; yet it should be able to backtrack if necessary.
5) There should be a mechanism for specifying and applying preferences, in relation to diverse linguistic phenomena.

The analyser constructed for AD-HAC adheres to these principles. It integrates its higher-level syntactic processes with its semantic processing, to produce a conceptual representation of the form illustrated in Chapter 2.

The main components of the analyser are:

a) An ATN whose task is to locate the basic building blocks of sentences: simple noun groups, verb groups, prepositional phrases, conjunctions and wh-forms.
b) A set of requests, classified into 5 types, associated both with individual words in the dictionary and with standard operations, such as relative-clause processing.
c) A control mechanism for applying these requests, with a small finite memory and a preference-based strategy for controlling a nondeterministic, predominantly left-to-right, analysis.

Section 5.2 describes these components in detail, using one of the dictionary entries by way of example, and also explaining the "small finite memory" requirement. Section 5.3 then describes this analyser's treatment of some linguistic phenomena, such as ambiguity, and that of some specific constructions such as relative clauses and conjunctions. Section 5.4 indicates some of the present shortcomings of the implementation, but suggests that no fundamental modifications to the analyser are needed to overcome these.

5.2) The basic components

Figure 5.1 below illustrates how the three basic components of the system are related: it is intended simply to provide some context for the following subsections, which describe in some detail the ATN, the requests, and the control mechanism.

```
------------------------------ Figure 5.1 ----------------------------

        Sentence
            |
            |
            V
            ATN <-------------------------| Word syntactic labels |
            |
            |
            V
      constituent tree
            |
            |
            V
      | Strategy box | analysis1 |
      | with limited | analysis2 |
      |  memory for  | analysis3 |  <-------| situation requests |
      |   partial    | analysis4 |
      |   analyses   | analysis5 |
      |              | analysis6 |  <-------| word requests |
      |______________| analysis7 |
            |
            |
            V
     conceptual analysis

---------------- The gross structure of the analyser ---------------
```

5.2.1) The ATN

An ATN is used here to isolate "constituents" of a sentence, rather than to build a complete structure which gives the syntactic relationships between all the constituents. The constituents it recognises are conjunctions, verb groups, simple noun phrases, prepositional phrases and wh-forms, all defined in a conventional way.

Conjunctions
Since the ATN is here merely isolating constituents, rather than building a representation of an entire sentence, the notorious problem of conjunctions is simply bypassed: the constituent analysis of the word AND is just "(conj AND)".

Verb groups
The ATN's analysis of verb groups is very simple, since all that is required later is access to the main verb, tense, voice, form and negation. Thus "WAS NOT BEING GIVEN" would be represented by the ATN as the constituent

(VP (tense PAST) (form PROG) (voice PASSIVE) (neg T) (verb GIVE))

Simple noun phrases
The phrase "simple noun phrases" is used here to exclude prepositional phrase modifiers, appositives and relative clauses: these are treated elsewhere. Simple noun phrases may include, for example, determiners, quantifiers and adjectives: for such noun phrases, the ATN will provide an analysis of the same gross form as the ultimate conceptual analysis of that noun phrase, omitting time-references and the resolution of ambiguous adjectives or nouns.

The characterisation of "SEVERAL OF FRED'S BIG BLUE BLOCKS", for instance, is almost precisely in the CD notation described in Chapter 2:

```
(NP (FOCUS (STATE THING))
    (STATE (STATENAME QUANTIFY)
           (VAL *SOME*)
           (THING (FOCUS (STATE THING))
                  (STATE (STATENAME POSS)
                         (VAL FRED)
                         (THING (FOCUS (STATE THING))
                                (STATE (STATENAME SIZE)
                                       (VAL 4)
                                       (THING (FOCUS (STATE THING))
                                              (STATE (STATENAME COLOUR)
                                                     (VAL BLUE)
                                                     (THING BLOCKS))))))))
```

Wh-forms are treated similarly, but using the label WHNP rather than NP; relative pronouns - WHO, WHICH, WHERE, THAT - are labelled RELNP.

Prepositional phrases

Prepositional phrases are treated as incorporating noun phrases, and may be keyed by PP, WHPP and RELPP as appropriate; the constituent corresponding to the phrase "NEAR SEVERAL OF FRED'S BIG BLUE BLOCKS" would be

```
(PP (prep NEAR)
    (noun (FOCUS (STATE THING))
          (STATE (STATENAME QUANTIFY)
                 ... etc
```

Even in this restricted application of an ATN, it must be noted that nondeterministic processing is still necessary, since strictly local ambiguity may still be present. To illustrate this, imagine that a constituent analysis must be found for the words "HER MONEY": and, for simplicity, imagine that the ATN consists of just an NP network. The two constituent analyses, for (1) "HER MONEY" as a single NP, and for (2) "HER" "MONEY" as two discrete NPs, must be found. More complex examples can be readily imagined; for instance, "HER SAW MILLS" where SAW and MILLS may both be either verbs or nouns, and where the concatenation "SAW-MILLS" may be considered.

The ATN, as mentioned earlier, is used essentially as a first-pass fragmentation routine to make groups of words easily handled by the later, main, stage of processing; and this later stage operates in a predominantly left-right fashion. The data structure produced by the ATN, designed to facilitate this later processing, is called a "constituent tree".

The structure of a constituent tree is recursively defined: it comprises a list of "branches", each having a single constituent - which will correspond to some substring of the sentence - followed by the constituent tree for the remainder of the sentence.

There are in fact several separate networks in the ATN, corresponding to noun phrases, verb groups, prepositional phrases and (trivially) conjunctions. All networks are tested against all substrings; and, if no network accepts a substring, the first word is treated as an isolated word, and the remaining substring considered by all networks.

This use of an ATN has three principal advantages. Firstly, it provides a convenient mechanism for treating words with more than one syntactic category, especially since frequently only one category will be considered because of the influence of the preceding context. Secondly, it provides a motivated and isolable fragmentation scheme, in contrast to that of Wilks, which can identify the important words in a sentence (verbs and head nouns), and thus allow groups of words to be treated as a whole, unlike Riesbeck's system. (A similar strategy has been adopted by Boguraev, as described in [Boguraev, 1979]) Thirdly, each element has an easily identified head word with which requests are often associated.

The utility of the ATN is improved by the addition of a test which may be used on the arcs leaving a state. This test, called PARSING-FAILED, can detect whether any previous arcs leaving a state have succeeded, ie reached a POP arc. The use of this test acknowledges that an ATN is fact depends sometimes upon the ordering of arcs leaving a state, and enables the ATN to exploit this dependence. It is used in two situations: firstly, to block the spurious recognition of verb phrases which actually are merely auxiliary to some other verb phrase, eg 'WAS' in 'WAS NOT BEATEN'; and secondly, to recognise that the content of a quantified noun phrase has been elided, eg in "I HAVE A FEW".

5.2.2) <u>The requests</u>

A set of requests, each having two predicate parts and one action part, is used to guide the analysis of sentences. They work on the constituent trees delivered by the ATN, to build up the conceptual analysis of the whole sentence. The first predicate of a request determines whether the action part is to be executed, while the second determines whether the request should be discarded if it is not used. These two predicates are named respectively the MAIN and the KEEP predicate. The requests are used to specify what should be expected,syntactically and semantically, of the remaining text, and to determine the utilisation of the expected items when they arrive.

The requests come from two sources: firstly from the definitions of words in the dictionary, principally from verbs and conjunctions; and secondly from places in the program where a standard situation has been recognised; for instance, the beginning of a sentence, relative clauses, or sentences of the form "x DO y TO DO z".

The requests fall into five request classes, each corresponding to a step in the processing of a constituent, which proceeds in a cyclic fashion described in the next section. Each request is labelled with its type. The five types are:

i) USE - use a constituent found by the ATN; if a constituent can be used, any program associated with the head word is executed. In the case of verbs, this loads in the requests which will process the rest of the sentence.

ii) TRIVIAL - perform any (unconstrained) action; all verbs load requests of this type whose purpose is to build up structures, insert time information and negation, and to add further requests.

iii) USE-REG - handle constituents which have been placed in "registers", a form of local storage. Most usually this is just for placing the subject into the structure built by a verb sense.

iv) UNEMBED - insert the structures corresponding to embedded clauses, relative clauses and prepositional modifiers, into a predetermined place in the clause above. Also resets structure, registers and the list of requests.

v) EMBED - save existing structure, registers and requests, and prepare to process an embedded clause of some kind.

The characteristics of these five types which make it useful to differentiate between them will be explained in detail in section 5.2.4, after the control mechanism and "theories", the objects the mechanism manipulates, have been described. Figure 5.2 is a grossly simplified illustration of the application of requests to the simple sentence "JOHN WENT TO THE PARK". The fragments "JOHN" "WENT" "TO THE PARK" have been found, and partially processed, by the ATN. The first four steps are part of a standard sequence for handling declarative sentences, and the remaining six are specific to the verb "GO". Each USE step is preceded by a GET, which is responsible for selecting an item from the constituent tree delivered by the ATN.

------------------------------ Figure 5.2 ------------------------------

```
      JOHN         WENT                          TO THE PARK
        |            |                                |
GET--->USE-->GET---->USE-->TRIVIAL-->USE-REG-->GET-->USE-->GET-->USE
     standard sequence       |          loaded        loaded      ends
                       the request        by            by      sentence
                       for "GO"         WENT          WENT
                     builds sem.str.
                     & loads requests
```

--------------- Simplified example of request application --------------

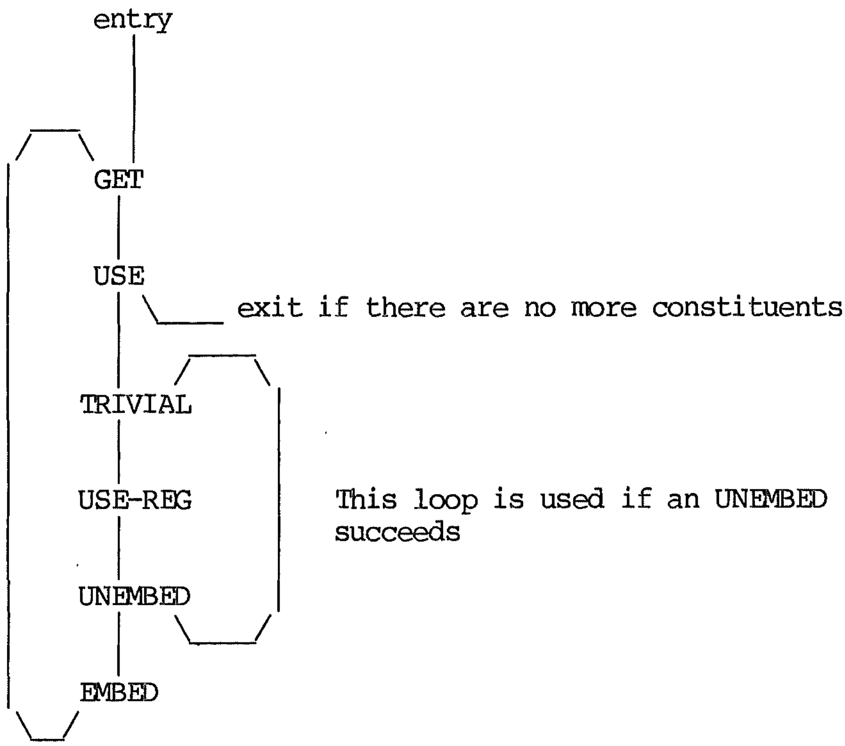
5.2.3) The control mechanism: "theories"; and memory limitations.

Based on the notion of preference, a control mechanism for the application of requests has been developed which, though in a non-deterministic environment, has the important property of needing only a small finite memory for partial analyses.

The objects with which this mechanism works, I call "theories": these correspond to partial analyses: they essentially "package" requests with environments. When tried, a theory will usually generate new theories. Each theory has seven components:

1) A numeric score, usually in the range -10 to 50.
2) A partial semantic structure.
3) A set of available requests.
4) An indication of the request-class to be selected.
5) The remaining constituent-tree (the product of ATN processing)
6) A current constituent if the request-class is USE
7) A set of registers, a form of local storage

The basic mechanism is simple, and is related to the Graph Traverser [Doran, 1965], Woods's "theories" [Woods, 1977] and the KRL scheduler queue [Bobrow et al, 1977]. The theories are maintained in order of preference, and the most-preferred theory is selected and evaluated: that is, those requests which are of the class indicated by the class tag (item 4 above) are tried. If the first predicate of a request succeeds, the action part is executed: then this request, and any other request of the same class whose KEEP predicate fails, is discarded, and a new theory generated. The generation of a new theory requires specification of a request class. This follows the pattern



The loop shown from UNEMBED to TRIVIAL applies only when an UNEMBED has been obeyed, permitting multiple unembeddings without consuming more constituents, and also permitting an UNEMBED to influence TRIVIAL or USE-REG requests directly (though this is seldom needed). When no UNEMBED is done, EMBED is selected next; or in the absence of EMBED requests, GET.

(GET does not label any requests at all, but is the stage of processing where a new constituent is found, usually from the constituent-tree returned by the ATN (but see section 5.3.2). With the exception of GET and USE, a theory will bear a given class-tag only if there are requests of that type in the theory; otherwise the class is chosen to be the next in the above cycle for which requests do exist, and failing that, GET.)

The actual mechanism is in two respects more sophisticated than this simplified account suggests: I have said that a theory is selected from the top of the stack of theories, on the basis of preference, but it may be that there are several theories of the same preference. These are all separated from the stack, and processed in turn. This avoids problems where one theory generates new theories with a slightly higher preference, causing the next theory, intrinsically of the same value, to be obscured.

The second difference has, I believe, important implications concerning the relation between memory and linguistic faculties. This is the observation that, despite the apparently non-deterministic operation of the analyser, the depth of its stack of theories need only be small; the size of stack needed compares with the supposed size of human short-term memory. Specifically, limiting the size of this stack to seven theories, and simply discarding any theories which fall off the end, actually improves the performance of the analyser for complex sentences, even elaborately embedded ones. Indeed, the largest stack needed for any sentence which has been processed so far (using the present grammar and vocabulary) is only 5 theories long. This result is an empirical observation, and does not depend on any fine-tuning of the preference manipulations. It may turn out that the addition of large numbers of highly-polysemous words may require adoption of a wait-and-see strategy, such as those used by [Hayes2, 1977] and [Marcus, 1980] However, though the dictionary does contain a number of polysemous words, no problems of this nature have been encountered so far.

[Marcus, 1980] proposes a "determinism hypothesis", and describes an analyser which also has a limited memory. This memory consists of a stack of partial syntactic structures, and the rules of the grammar are permitted to inspect, using look-ahead, only a small number (usually 3) of these structures. However, Marcus does admit that his analyser is unable to deal with complex sentences, and must seek advice from some other, more powerful, component of the larger system. It seems to me that this is tantamount to accepting that syntax rules are inadequate when so constrained; and his analyser does not in fact attempt to provide the required additional semantic analysis. My approach includes semantics.

To close this section on an optimistic note, let me make an untested proposal. A vexing problem, to which I believe the concept of a limited memory provides a partial answer, is the problem of determining when a sentence is structurally ambiguous. There seems to be a highly plausible rule: Only if, working within the memory limitations of the analyser, more than one analysis can be found, and later analyses violate no more noun-feature preferences than did the first analysis, then that sentence is ambiguous.

5.2.4) The stages of processing: details

The stages of processing, excepting GET, are conveniently described by reference to the classes of request to which they correspond, since these stages form the the basis of the classification.

The requests themselves, as mentioned in section 5.2.2, always have a MAIN predicate, a KEEP predicate, and a set of actions. A theory has a type which - excepting GET - selects a subset of the requests for evaluation. When a request is evaluated, the resulting theory (if any) does not receive a copy of that request, nor of any other requests of that type whose KEEP predicates fail.

i) GET

GET is the first step in each cycle; it is exceptional since it labels no requests. GET is discussed more fully in section 5.3.2 below, where the treatment of questions and relative clauses is examined: for present purposes, I simplify by saying that GET retrieves an item from the constituent-tree given by the ATN; for each retrieved item, a new USE theory is constructed, with the appropriate "current constituent" and "constituent-tree".

ii) <u>USE</u>

USE requests have a MAIN predicate which normally specifies the syntactic class expected. Their action parts, in addition to specifying the use to be made of the constituent they accept, will often contain preference manipulations based upon the features associated with nouns: for this purpose, features are arranged in a hierarchy, which was given in Chapter 2.

For each sense of the head word of a constituent, each USE request is applied. (USE requests may branch in two different ways; for different senses of words, and for different ways of using the constituent.) For each successful request application, that is, where the MAIN predicate succeeds, a new theory is created, with a class-tag determined according to the rules given in section 5.2.3 above.

If a constituent is encountered for which no request succeeds, it is tested to see if it can be "trapped"; traps handle conjunctions, and purposive TO-complements, as in "I WENT TO THE KITCHEN <u>TO</u> <u>MAKE</u> <u>SOME</u> <u>COFFEE</u>". Traps are discussed more fully in section 5.3.3 below.

iii) <u>TRIVIAL</u>

TRIVIAL requests are simple - hence the name - and cause the creation of only one new theory. Any number of such requests may be present; the MAIN predicate of each determines whether the action should be performed but the KEEP predicate is a vacuous formality.

TRIVIAL requests are extremely common, being the vehicle for the introduction of other requests from the dictionary. These requests are often used to build conceptual structures, add time and negation information, and load the other requests which will analyse the remainder of a clause.

iv) <u>USE-REG</u>

USE-REG requests take a constituent, usually the subject, from temporary storage in a register. Since the constituent has been placed there by a USE request, it corresponds to only one sense of the head noun. However, it sometimes happens that more than one USE-REG request exists, and in this case all the available USE-REG requests are applied, producing one new theory each. This situation occurs mostly for the passives of indirect-object verbs, where the subject may fill either the slot corresponding to the direct object or the indirect object in an active sentence.

v) <u>EMBED</u>

EMBED requests deal with embedded clauses, relative clauses and prepositional phrase modifiers. An EMBED request is usually optional; one theory will be created just as though there had been no EMBED requests present. For each EMBED request whose MAIN predicate has succeeded, a distinct embedded theory will be created. These theories will have the current state of the registers, the current set of requests, and the current semantic structure, all saved on an otherwise blank list of registers. When this has been done, the actions of the applicable EMBED request are executed. These will always add new requests to deal with the embedded clause, and will often set a few registers. For the latter purpose, there is temporary access to the saved registers, which is necessary to handle the tensing structures which occur in embedded clauses, and those clauses which borrow the subject from a higher-level clause.

An obvious point is that an analysis cannot be allowed to terminate while still in an embedded state.

vi) UNEMBED; and constraints.

UNEMBED requests are the converse of EMBED. There can only be one UNEMBED request present at any time. Requests of this class restore the registers, requests and semantic structure before their action parts are executed.

An UNEMBED request is also optional; thus one theory will always be created which corresponds to a failure to unembed. If an unembedded theory is created, its preference is modified according to the following rule: If there are any USE requests which are being lost by unembedding, decrease by 3; otherwise decrease by 1. This means that there is always a tendency to remain embedded. The strategic reasons for this are discussed in section 5.2.6.

However, the MAIN predicate on the UNEMBED request is not regarded as sufficient to permit an unembedding. A further mechanism, the specification of constraints, is involved. A general constraint on unembedding is that there must be a semantic structure which has been built. Certain verbs, however, specify further constraints - saying such-and-such a request must be used. If any of these constraints have not been satisfied, no unembedding can take place. (This mechanism applies also to accepting the end of the sentence, which is seen as a special case of unembedding.)

It has also been found useful to allow requests to specify certain operations to be performed at the end of a clause; this is done by means of the two functions BEFORE-UNEMBED and AFTER-UNEMBED, which load small programs which are executed just before, and just after, the process of restoring registers, requests and semantic structure: or, for the main clause, at end of sentence.

Taken together, these requests constitute an effective mechanism for driving a semantic analyser which can exploit syntactic information. The addition of a KEEP predicate to the simple test-action requests Reisbeck described neatly solves the problem of discarding irrelevant requests, and the nondeterministic framework in which requests are activated takes over the task of selecting amongst possible analyses. The criticisms of Reisbeck's work, that the requests were neither manipulable nor clean, do not apply to this work. However, the requests can become rather complex: the next section describes how this complexity is handled by the use of macros. With this addition, the requests acquire elegance without sacrificing effectiveness.

5.2.5) The dictionary entries

The analyser is controlled to a high degree by the definitions of the words in sentences, and particularly those of verbs. These definitions are encoded as requests, which (apart from the auxiliaries BE,DO,HAVE) follow the same general pattern, being TRIVIAL requests which:

a) Set up a semantic structure
b) Make various declarations about that structure
c) Put time information into the structure
d) Create new tokens to fill certain slots of that structure
e) Add a set of requests to handle the rest of the sentence (or clause).

The dictionary entry for DISPLEASE, a verb with only one sense, is shown below. It should become apparent that the verb definitions are equivalent to the specification of case-frames; it may be convenient to consider the analysis process as following a dynamically-created ATN.

```
(DISPLEASE
  ((requests
      (ADDREQ
         (TRIVIAL T NIL
            (BUILDS
                 ((CAUSE
                     (ANTECEDENT
                        (EVENT (ACTOR DUMMY-HUMAN1)
                               (ACT DO)))
                     (RESULT
                        (STATE (STATENAME JOY)
                               (THING DUMMY-HUMAN2)
                               (VAL (LOWERBY 2)))))))
            (TIME-PLACE)
            (VERBS-DUMMIES
                 ((CAUSE ANTECEDENT EVENT ACTOR) DUMMY-HUMAN)
                 ((CAUSE RESULT STATE THING)     DUMMY-HUMAN))
            (COND
               ((ACTIVE)
                   (PLACE-SUBJ (CAUSE ANTECEDENT EVENT ACTOR) ANYTHING)
                   (PLACE-OBJ (CAUSE RESULT STATE THING) BEAST
                      (PLACE-SUBJECTLESS-CLAUSE BY (CAUSE ANTECEDENT)
                         (AND (TENSE-IS PRES) (FORM-IS PARTICIPLE))
                         subj)))
               ((PASSIVE)
                  (PLACE-SUBJ (CAUSE RESULT STATE THING) BEAST )
                  (PLACE-CLAUSE BY (CAUSE ANTECEDENT)
                     (AND (TENSE-IS PRES) (FORM-IS PARTICIPLE)))
                  (PLACE-PP BY (CAUSE ANTECEDENT EVENT ACTOR) ANYTHING)
                  (PLACE-SUBJECTLESS-CLAUSE BY (CAUSE ANTECEDENT)
                     (AND (PASSIVE) (FORM-IS PROG))
                     subj)
                  (PLACE-SUBJECTLESS-CLAUSE AT (CAUSE ANTECEDENT)
                     (AND (PASSIVE) (FORM-IS PROG))
                     subj))))))))
```

In this example, there is only one definition for the verb; 'requests' labels the part of the definition which is to be executed. Here the function ADDREQ is used to add a request. The request is of class TRIVIAL, and its MAIN predicate is T: therefore it will succeed. Its actions are:

BUILDS to build a semantic structure
TIME-PLACE to insert temporal information in a standard way
VERBS-DUMMIES to create new 'tokens' to fill in parts of the structure: this corresponds to assuming features of objects in default.

Both ACTIVE and PASSIVE are functions which inspect registers which have been set up at the time the verb group was encountered.

As requests can be fairly complex objects, involving two predicates and an arbitrary set of actions, their full specification on each verb definition would be a tedious and repetitive process. Advantage can be taken of the repetitiveness of simple requests however, by defining a set of macros with which they can be built up. These macros expand into full requests while the analyser is running. Several such macros have been provided, and very few verbs need facilities which these macros do not provide. However, when such facilities are needed, it is still possible to write special-purpose requests. The existence of the macros does greatly ease the definition of new verbs. The macros are:

PLACE-SUBJ for using the subject: generates a USE-REG request

PLACE-OBJ for using a noun-phrase: generates a USE request

PLACE-PP for using a prep-phrase: generates a USE request

PLACE-CLAUSE
PLACE-SUBJECTLESS-CLAUSE look for various sorts of embedded clause
PLACE-OBJECTIVE-CLAUSE

GET-PARTICLE looks for a specific word, and is capable of splitting it from any component with which it is associated in the constituent-tree.

In the ACTIVE branch for DISPLEASE,

(PLACE-SUBJ (CAUSE ANTECEDENT EVENT ACTOR) ANYTHING)

adds a USE-REG request which will look at the 'subj' register, and place the conceptual representation of the subject at the end of the path (CAUSE ANTECEDENT EVENT ACTOR); also it will apply, in this case, the undiscriminating preference for the feature ANYTHING (since ANYTHING is very near the top of the feature hierarchy described in chapter 2).

The second macro call in the ACTIVE branch,

(PLACE-OBJ (CAUSE RESULT STATE THING) BEAST ...),

adds a USE request which seeks a noun phrase, preferring the feature BEAST to be present: when this has been found,

(PLACE-SUBJECTLESS-CLAUSE BY (CAUSE ANTECEDENT) X...X subj)

adds a further request, which corresponds to an EMBED. This will only be used if the word BY is located, in which case an embedded clause is expected, which will lack a subject, and whose verb is a present participle. The current contents of the register 'subj' (ie whatever did the displeasing) is to be used again as the subject of the embedded clause. The conceptual representation of the entire embedded clause is to be placed at the end of the path (CAUSE ANTECEDENT).

In the PASSIVE branch, there is again a PLACE-SUBJ macro call, and two PLACE-SUBJECTLESS-CLAUSE macros which differ only in the word they expect as a cue. The two new items are:
i) (PLACE-CLAUSE BY (CAUSE ANTECEDENT) X...X),
which looks for a complete clause cued by the word BY, whose verb is, again, a present participle. This will correspond to sentences like

"FRED WAS DISPLEASED BY MARY GOING TO THE SHOPS"

ii) (PLACE-PP BY (CAUSE ANTECEDENT EVENT ACTOR) ANYTHING)

creates a USE request which looks for a prepositional phrase, with the preposition BY. This will correspond to sentences like

"FRED WAS DISPLEASED BY MARY"

5.2.6) Preference manipulations

Preference, the numeric score associated with a theory (and typically in the range -10 to +50), is important because it provides the basis for the selection of theories in a nondeterministic environment. The idea of preference is taken from Wilks; but the implementation in AD-HAC is rather different. It was noted in section 5.2.3, in connection with the observation that only a small stack is needed for theories, that this small memory requirement does not depend upon finely tuning the preference manipulations. Preference is however an important device, and is manipulated in several situations:

Feature matches and mismatches

These correspond in spirit to Wilks' preference mechanisms; however, whereas Wilks uses direct matches between semantic primitives, with the possibility of matching against classes, I have found it useful to perform two distinct types of matching operation. Firstly, there are "primary features", which are organised in a hierarchy, and which were discussed in Chapter 2: a request which "prefers" feature X will accept a noun-sense with feature Y if Y is anywhere on the branch of the hierarchy headed by X. (By "accept a noun-sense", I mean the preference will be applied.) Secondly, there are "secondary" features, such as MASSY, PROPERNAME and RELATION, which do not occur on the hierarchy, and with which only direct matches are considered.

Normally, feature preferences are applied positively: if there is a match, the 'score' of the next theory generated is increased by 2. Simultaneously, all USE requests decrease the score of the next theory by 2. This has two effects. Firstly, when a noun sense has the expected features, the new theory will have the same preference as the original (and so will still be at the top of the stack), but if the expected features are not present, the preference will drop; any competing sense of that noun will be preferred (preference only makes sense when dealing with competing alternatives). Secondly, USE requests which pick up verb groups do not usually apply any preference: therefore the subsequently formed theory will have a lower score than its parent, and will possibly not be at the top of the stack any more; other analysis paths which are sufficiently close will now have a chance to proceed, and may perhaps increase their score. This is a helpful feature, because different analysis paths may correspond to different syntactic expectations, and may not be in step with one another; this mechanism helps to make the processing of such alternatives more uniform.

Embedded clauses, and unembedding.

When an EMBED request may be applied, two theories are created which correspond to embedding and failure to embed, and these have the same preference. However, the macros from which these requests are usually generated specify a number of USE requests on the lower level, and the first of these will, if successful, increase preference by 2. The effect of this is to stimulate an embedded path which has located a component which it needs.

It has been noted (section 5.2.4) that UNEMBED requests will decrease the preference of the unembedded analysis, by 3 if some other component is expected, by 1 otherwise. The general disinclination to unembed reflects the observation that ambiguously placed conjunctions are most often taken to conjoin clauses at the most deeply embedded level. For instance, a sentence given by [Cullingford, 1978] (which he quotes from a newspaper) illustrates this point nicely:

"A NEW JERSEY MAN WAS KILLED FRIDAY EVENING WHEN THE CAR IN WHICH HE WAS RIDING SWERVED OFF ROUTE 69 AND STRUCK A TREE"

Cues

There are some words, such as BILL and JOHN, which may be either simple nouns or proper names. The use of names is notoriously complex [Carroll, 1979]; AD-HAC takes a very simple, and probably inadequate, view. The use of a determiner, or of a possessive construction, is taken as a cue to the sort of noun expected: if one exists, then the noun is expected to be a simple noun; if not, a proper name. In general, the selection of a noun sense is penalised if these cues are inappropriate. An exception occurs when a relative clause follows, for instance "THE BILL TO WHOM I SPOKE". In such a case, no penalty is applied, and the usual feature-matching procedures must work unassisted.

Overwriting

Once a constituent has been placed into the semantic structure, any attempt to replace that part of the structure will be penalised; in practice, this permits verbs to be more easily defined.

5.3) Linguistic phenomena: further details

The preceding sections, 5.1 and 5.2, have described the principles behind the design of the analyser, and the basic components of the implementation. This section explores the application of the analyser to its task, namely the representation of the meaning of sentences, by considering several features of language use, and describing how the analyser accounts for them. I consider a global phenomenon, ambiguity; and then particular features: particular syntactic structures, relative clauses, questions, conjunctions, negation, and tensing structures.

5.3.1) Treatment of ambiguity

The analysis model described above attempts to handle four kinds of what might be loosely termed ambiguity.

i) Single words in multiple syntactic classes, related in sense or not

An explicit syntactic analysis, whether of complete sentences or, as here, of well-formed constituents, goes a long way towards handling this form of ambiguity. In this model, the ATN performs this function, but sometimes is unable to determine the correct constituent. The application of requests, which correspond to expectations, will in these cases make a decision on the basis of surrounding context.

ii) Multiple senses of a word, with a common syntactic category

The treatment of this phenomenon differs according to the syntactic category.

Verbs with more than one sense are normally distinguishable on the basis of their case-frames, though occasionally there is an interaction with the features of surrounding nouns. Already, in my (pathetically) small dictionary, there is one example that cannot be determined by this technique: this is the choice of GET = FETCH or GET = ACQUIRE; I have had to leave this as an open problem. This is clearly unsatisfactory.

Nouns with several senses are normally distinguishable by their features, but occasionally CUES are useful: for instance, proper names are not normally introduced with determiners. This information is communicated to the overall request mechanism by inspecting CUES in the USE phase; this causes behind-the-scenes preference changes.

Prepositions usually have a large number of senses; each sense is however applicable only in certain syntactic contexts. The most common use of prepositions is to indicate that a following noun phrase fulfils some role in the case-frame associated with a verb; prepositions serving this function are picked up by the use of PLACE-PP macros in the verb definition, and any independent sense the preposition may have is irrelevant. For use as postmodifiers, however, prepositions do need definitions, and these are kept as requests associated with the preposition.

Adjectives may have many senses; the current program does not attempt to disambiguate these, but simplistically assumes each adjective has one sense only.

Despite the weakness with adjectives, this treatment of lexical ambiguity seems quite good.

iii) Structural ambiguity of sentences

The analyser presented here is intended as a front-end to an inference mechanism, and attempts to find the most plausible reading of a sentence. A sentence is reanalysed only if the inference component finds that the first reading found is ludicrous. In this case, the analyser will repeat its analysis, but reject the first reading encountered. If the next reading is also found to be ludicrous, the analyser will repeat, and reject both of these. And so on, until either the inference component finds an acceptable reading, or the analyser can find no more readings.

This is not a long-term solution to the problems of genuine structural ambiguity; but the discussion in section 5.2.3 may point the way.

iv) Referential ambiguity

Referential ambiguity is not tackled by the analyser itself, but its resolution is one of the principal tasks of the inference mechanism, and will be discussed in Chapter 8. The part played by the analyser in the resolution of referential ambiguity is twofold: for pronouns, it must provide information to the inferencer which will permit it to determine whether other nominals could or could not be the referent of the pronoun; and for definite references, it must flag the corresponding tokens in such a way that the fact of definite reference is accessible to the inferencer.

The analyser's treatment of ambiguity is clearly quite comprehensive, though there remain areas where it is not as strong as might be wished. The simplistic treatment of adjectives is probably the weakest spot of all, and work is in progress on this - as on so much else!

5.3.2) Postponement mechanisms: Questions and Relative clauses.

Yes/no questions, and wh-questions which do not focus upon the subject, both exhibit the splitting of an auxiliary from the main verb group. For example, "DID MARY ANNOY JOHN?", "WHO DID MARY ANNOY?". Even a sentence like "WAS IT JOHN WHO ANNOYED MARY?" can be seen as an example of this phenomenon, if we allow "WAS" to function as both a fronted auxiliary, and as the entire main verb phrase. This general phenomenon is handled by modifying the basic behaviour of GET as described below.

Relative clauses are commonly regarded as containing a "trace" of the relativised object which belongs in, but has been elided from, the sequence of words forming the relative clause. Wh-questions are similar in this respect. This phenomenon is also treated by a modification of GET.

Section 5.2.4 simplified the operation of the GET phase by omitting mention of these two aspects, and presented the basic model which involves taking the next set of syntactic constituents, and creating new theories corresponding to each one. The GET phase in fact does more than this. Its more complex operations are controlled by the settings of registers (local to an analysis path), and include

i) Inserting an auxiliary to make a complete verb group
This is done for yes/no questions and some wh-questions. The auxiliary is taken from the register in which it is stored, and placed at the beginning of the remaining words of the sentence. The substring so formed is reanalysed by the ATN constituent grammar.

ii) Replacing a noun phrase or prepositional phrase constituent
This is common to relative clause and wh-question processing, and involves simply generating a new USE theory which must use the constituent. Wh-words in relative clauses are replaced by the selected sense of the relativised object. Examples of the situations where this replacement operation may be done occur in the sentences:

WHO DID JOHN ANNOY?
WHO ANNOYED MARY?
THE PERSON JOHN ANNOYED HIT HIM.
THE BANANA WHICH JOHN GAVE TO MARY WAS ROTTEN.
THE PERSON TO WHOM JOHN GAVE THE ROTTEN BANANA BECAME SICK.

Sometimes a relative clause, or wh-question, is deeply embedded. By this, I mean that a further clause is embedded within the relative clause (or question), and the "trace" of the relativised object is to be found within this embedded clause; for instance,
"THE CAR YOUR BROTHER SAID HE WAS EXPECTING US TO TELL JANE TO BUY"
(an example from [Winograd, 1971], which he calls "downrel") This often happens with verbs, such as 'SAY', which normally expect a clause optionally introduced by 'THAT'; however when the embedded clause contains a "trace" of a relativised object, 'THAT' seems quite out of place. Consider:

FRED SAID THAT THE MONKEY ATE THE BANANA.
*THE BANANA FRED SAID THAT THE MONKEY ATE WAS ROTTEN.

FRED SAID THE MONKEY ATE THE BANANA.
THE BANANA FRED SAID THE MONKEY ATE WAS ROTTEN.

When such an embedded clause is processed, the semantic structure will indicate the position of the relativised object (or queried object) within itself, and this must be combined with the position of the embedded clause to give a true representation.

It is necessary when processing relative clauses to make sure that quantifiers, if present, are properly scoped. In the representational language used here, this requirement is easily and simply met by placing the semantic representation of the relative clause inside any existing qualifying information.

iii) Combining a noun phrase constituent with a dangling preposition.
This operation is very similar to the straightforward replacement outlined above, but is performed (additionally) if the next word in the remaining part of the sentence is a preposition, and the constituent to be replaced is not a prepositional phrase (eg "TO WHOM").

5.3.3) <u>Traps: dealing with the unexpected</u>.

It has been explained that verb definitions provide a set of case-frames, which denote the expected gross syntactic elements of a clause. In an expectation-driven system, conjunctions present a special problem: it seems unreasonable to "expect" conjunctions, because they occur almost anywhere; yet if they are not expected, how can the system handle them when they occur?

In AD-HAC, conjunctions determine themselves how they are to be used. They are equipped with "traps", which, like requests, have a predicate and an action. The predicate part of these traps will normally test whether a clause can be terminated; and the actions will normally

a) remove all existing requests except UNEMBED requests
b) Build a new semantic structure, having the existing structure as a part
c) Forbid further conjunctions from operating at this level
d) Initiate the processing of a new clause, EMBEDded into the newly-built structure.

The treatment of AND, BECAUSE, BEFORE and AFTER follows this outline. However, BECAUSE, BEFORE and AFTER may also begin a clause; for instance
"BECAUSE MARY WENT TO THE PARK, JOHN FELT UNHAPPY"
These conjunctions therefore have several traps, whose predicates distinguish the particular cases. The action-parts for these senses will also consume a comma if it occurs at the end of the first clause.

Similarly, purposive TO-complement constructions are handled by a trap; again, there is a test to ensure that the first clause is complete, and other requests are discarded.

Commas, apart from those expected at the end of certain classes of clause, are handled by a trap. This trap looks at the next constituent, to see if it too will be handled by a trap. Thus conjunctions may always be preceded by commas, as may purposive TO-complements.

5.3.4) The handling of negation.

AD-HAC's analyser recognises two ways in which a sentence may be negated: by the use of the word 'NOT' (or DIDN'T, WASN'T etc.); or by use of a phrase like 'NOBODY' or 'NONE OF THEM'.

The representation built for a negated clause differs in two respects from that built for a straightforward affirmative clause. Firstly, the conceptual structure has "(TRUTH FALSE)" instead of "(TRUTH TRUE)", or vice versa (though even this is not straightforward in the case of causal structures: see 2.3.1.3); secondly, some of the tokens appearing in that structure will be flagged as "existentially-qualified", which indicates to the inference mechanism that no referent for them should be expected. The significance of this flag to the inferencer will be discussed in section 8.5.1.

When either of the two sources of negation is found, a register is set to record that the current clause must be negated. From that point on, any noun phrase with the quantifier 'ANY' is flagged in the above manner; if the negation is introduced by the use of 'NOBODY' or similar, the corresponding token is also flagged in this way; and if any of the tokens introduced by VERBS-DUMMIES remain in the conceptual structure when the analysis of the clause is complete, they too are flagged.

Negation is a complex topic, and has received a great deal of attention from linguists. The treatment outlined above is offered as an engineering solution to a complex problem.

5.3.5) Tensing structures

The interpretation of tensing as reflecting relationships between the various points or intervals of time mentioned in a text is considered as two separate problems, intra-sentential and inter-sentential, and different mechanisms are used. Before discussing these separately, a brief reminder on the representation of time is in order.

"Time tokens" represent either points in time, or intervals. Time intervals use time points to denote their start and end points, and these may be definite or indefinite. The relations BEFORE and AFTER may hold between time points; and the relations DURING and -DURING provide a shorthand for relating intervals to time points or to other intervals. Some examples of the structures which may be generated are:

```
(TIME (NAMED *NOW*))

(TIME (NAMED TIMEPOINT1)
      (COMPARISON (BEFORE *NOW*))
      (COMPARISON (AFTER TIMEPOINT2)))

(TIME (NAMED TIMEPOINT1)
      (COMPARISON (DURING (NAMED TIMESPAN1)
                          (TS TIMEPOINT2)
                          (TF *NOW*))))

(TIME (NAMED TIMEPOINT3)
      (COMPARISON (BEFORE (NAMED TIMEPOINT4)
                          (COMPARISON (BEFORE *NOW*)))))
```

The first example illustrates the simplest possible time reference, while the second and third examples show two ways of relating the times TIMEPOINT1, TIMEPOINT2 and *NOW*. The third example does this by saying that TIMESPAN1 embraces TIMEPOINT1, and has TimeStart = TIMEPOINT2 and TimeFinish *NOW*. The last structure is one such as might be generated to represent the time reference of the sentence "FRED HAD GONE TO THE PARK", where TIMEPOINT4 is called an "intermediate time token", because it is implicitly referenced and mediates between two "real" points. Such intermediates play an important role in linking complex time references, both between sentences and within sentences.

<u>Inter-sentential relationships.</u>

The most basic observation here is that sentences whose main verbs have the same tense can be placed in a simple sequence. The use of complex tenses, such as past perfect, indicates that the time of this sentence is related in some other way to the time of preceding sentences. In the case of past perfect in a predominantly simple-past narrative, for instance, the past-perfect is earlier than the prior simple-past. For example,

JOHN WENT TO THE PARK. HE HAD QUARRELLED WITH MARY.

For such cases, the "intermediate time token" is identified with the main time token of the previous sentence.

Questions, however, cannot be related to previous sentences at all; the determination of time references for questions must be performed by inference, and is discussed in Chapter 8.

Unfortunately, there does appear to be a class of verbs for which these general rules are invalid, which is illustrated by the example:

JOHN ATE A BANANA. IT TASTED GOOD.

I have not pursued this problem.

<u>Intra-sentential relationships.</u>

The analyser described here attempts to use the information conveyed by tenses and tense shifts, and to assign time tokens to the various semantic structures which represent the clauses and phrases of a sentence. However, this sub-problem is beset with many difficulties, and the analyser is not entirely successful in this respect.

Given that time tokens refer either to points in time or to intervals, and that BEFORE, AFTER and DURING relationships may be represented, the problem is to determine, using the tense, and occasionally the form, of verbs in clauses, the relationships between the times to which they refer.

The mechanism used here involves the use of two "registers", which I call <u>this-time</u> and <u>curr-time</u>. "this-time" corresponds to the time token for an individual clause, and "curr-time" indicates a gross temporal environment. At the beginning of the processing of a sentence, both are set to NOW. When a verb is located, "this-time" is replaced with a new time token, and the tense of the verb is used to determine the relationship between "this-time" and "curr-time". When an embedded clause is to be processed, "curr-time" for this clause is set to the time token held in "this-time" for the major clause, and the tense of the verb of the embedded clause is used to relate these two times in much the same way.

There are many phenomena which cannot be handled by such a simple technique, and which unfortunately require divers modifications to the basic scheme. These I discuss individually, with the aid of examples. It will be noted that several unsolved problems remain in this domain.

a) JOHN EXPECTED MARY TO GO TO THE PARK.
This is an example of a verb, EXPECT, which explicitly relates the times of main clause and subordinate clause. To handle this, it is necessary to set up "curr-time" for the embedded clause to be AFTER "this-time" for the main clause. This is done by using the function PASS-TIME in the call to the request macro PLACE-SUBJECTLESS-CLAUSE; this is done very easily in the AD-HAC framework, and this ease is one of the strongest points in favour of the request-based approach to parsing.

b) JOHN KNEW MARY WAS AT THE PARK or JOHN KNEW MARY WAS HITTING BILL.
In the case of embedded clauses which have simple tenses and either denote states or have a verb in progressive form, the tense of the verb is ignored and the time token used is the same as that for the major clause. The mechanisms for recognising these situations and taking appropriate steps are built into the basic time-manipulating functions.

c) DID JOHN GO TO THE PARK? or JACK CAME TUMBLING DOWN.
In section 5.3.2, it was noted that yes/no questions cause the auxiliary to be reunited with the rest of the verb group. Thus in the first of these examples, the constituent 'DID GO' appears. This constituent will have a tense, and it is this tense which must be considered rather than the tense of 'DID' alone; as may be seen in the sentence "HAS FRED GONE TO THE PARK?". Since no semantic structure is built until the main verb is encountered, these cases present no trouble. However, in the case of "JACK CAME TUMBLING DOWN" the roles are reversed: TUMBLING provides the semantic structure, but the time information is derived from CAME alone. The problem here is to suppress the effect of the tensing of a verb; and this requirement is met elsewhere, with the conjunction WHEN, and also with modal verbs, such as START and STOP. Therefore, rather than overcome the problem by arranging that the ATN grammar should recognise 'COME TUMBLING' as a separate constituent, the functions which manipulate time may be suppressed, by setting a special register.

d) THE BANANAS YOU GAVE ME WERE ROTTEN.
When a relative clause modifies the subject of a sentence, or of any subordinate clause, the basic problem is that the time token with which the relative clause is to be compared is not yet known. Where all verbs have a simple tense, it is normally impossible to determine the temporal relationships except by inference, but in those cases where a complex tense is used, the implied intermediate time should be identified with another time-reference in the sentence. The analyser is currently unable to do this.

e) MARY WAS HITTING BILL.
Progressive form, except in cases covered above (subsection (b)), usually indicates that the time referred to was an interval. This is not true for this example; rather the influence here is to indicate repetition. This problem has not been studied.

f) THEY HIT THEIR CHILDREN.

The verb HIT is rather special, insofar as its stem is identical with both the simple past form, and with the past participle. The example sentence illustrates the difficulty of determining time reference in such a case, since, in isolation, this sentence may be interpreted as reporting either an isolated event or a habitual activity. The general use of present tense to denote habit is not handled by the analyser.

g) MARY WAS GOING TO HIT BILL.

Reference to future times, even future with respect to some past time, is seldom straightforward. Often its use implies either intention, as in the above example, or that an event is "on the way". In either case, it often implies that the predicted event did not happen or could be averted. The analyser does not currently represent this information.

The interpretation of tensing patterns in AD-HAC is based on a set of standard rules; when these rules break down, the inherent flexibility of the request-based parsing approach comes to the rescue by permitting non-standard operations to be performed. This may be clearly seen in case (a) above: the request macros provide a set of actions for standard syntactic contexts, but these may be viewed as default actions which are easily overridden by providing extra arguments to the macros. By exploiting this flexibility, a wide range of phenomena can be easily handled.

5.3.6) Suggestion of inferences.

The 'ideal' NL system would have a fully integrated analysis and inference system: AD-HAC does not, but rather tries to factor out the linguistic aspects of understanding from the inferential aspects. One of the areas where this factorisation can be problematic concerns the use of words, or turns of phrase, which carry implications.

The point may be illustrated with two examples:

(i) "DIDN'T JOHN GO TO THE PARK?"

carries, over and above its literal interpretation as a yes/no question, the implication that the speaker supposed that John had gone.

(ii) "JOHN TOOK MARY A BOOK" vs. "JOHN TOOK A BOOK TO MARY"

suggests that John then gave the book to Mary.

While it is not clear that the unification of linguistic and extra-linguistic information would be of much help in accounting for these phenomena, it remains true that the separation of analysis from inference poses problems in cases such as these: the analyser, having access to the actual words used, should present the inferencer with a representation of the meaning of the words; but in these cases, there are two meanings, and furthermore they are meanings of different kinds. Simple conjunction of the various meanings is inadequate, at least when the representation language has no notion of "probably true".

The approach adopted here is to provide the analyser with a broader channel of communication with the inferencer. In addition to passing across a CDform which encodes the literal meaning of sentences read, it may also suggest inferences. Inferences suggested in this fashion are not yet used by the inference mechanism, but it is anticipated that they will have much the same status as the regular conceptualisations representing sentence meaning, and that the existing suggestion mechanism will have to be extended to associate probability ratings to the inferences suggested.

5.4) Shortcomings, and further development.

As noted at various points in the body of the chapter, my analyser is not capable of handling the full range of English. It is my belief that its defects could be corrected without making fundamental alterations to the system; but naturally this cannot be proved. Below, I discuss some of the more obvious areas where further work needs to be undertaken.

i) Constraints on pronoun reference, and reflexive pronouns.

In recent years, there has been a great deal of work in linguistics aiming at deriving rules which would reproduce judgements of acceptability of sentences where some pronoun is to be read as coreferential with some other noun phrase in the same sentence. For instance, [Culicover, 1976] has proposed a set of constraints, expressed in terms of syntactic structures within the transformational paradigm, upon the possible referents for pronouns. Though the analysis program described above does not build a syntactic representation of the whole sentence, much syntactic information is implicit in the pattern of embedding and unembedding; and this information would not need to be made explicit in order to derive the constraints he proposes.

Pronoun reference is not carried out by the analyser at all, but by the inference mechanism which inspects the semantic structures produced by the analyser. It is clearly desirable for the analyser to give further information, where applicable, about those referents which should not be considered at all.

Similarly, the analyser should inform the inference mechanism that a reflexive pronoun must be coreferential with some other token in the same sentence. Where there is only one compatible token, the analyser could simply replace the token for the reflexive pronoun with the other token; but there are cases where several such tokens exist, and therefore the inferencer must resolve the token.

The two problems, of mandatory coreference and of forbidden coreference, would require a further broadening of the channel between the analyser and the inferencer, in addition to extension of the analyser to decide what extra information should be passed.

ii) Participial premodifiers.

At present, phrases such as "A SWINGING LAMP" or "A WASHED CAR" are not handled. Their analysis would not, I believe, involve the addition of any new mechanisms; embedding and unembedding, together with the application of constraints, would suffice for this purpose.

iii) Abstract nouns.

Abstract nouns must be represented in the same manner as the verbs to which they correspond: similar conceptual structures must be built to represent, for example,

FRED'S DEATH .....
FRED DIED.

FRED'S DECISION TO ASK MARY FOR THE BOOK .....
FRED DECIDED TO ASK MARY FOR THE BOOK.

THE REVELATION OF THE COMPANY'S FINANCES .....
THE COMPANY'S FINANCES WERE REVEALED.

Since nouns may carry requests in the same way as verbs do, the analysis of abstract nouns could be implemented in the same manner; the principal problem is that, since structures are to be built rather than simple tokens, strong constraints must be applied when they are placed into the semantic representations built. Also, the presence of a structure of this sort will often influence the interpretation of subsequent words. Whilst these interactions are the raison d'etre of the request mechanism, their variety is daunting. Nevertheless, it is planned that the analyser will be able to handle abstract nouns in the future; and it is expected that the philosophy of the analyser's construction will remain unchanged.

iv) "HOW" questions.

Questions "how" may be analysed in different ways. In terms of the semantic representation used here, they may relate to the "instrument" role of events, or to the "antecedent" role of any "cause"; or to "manner". Except by knowing the answer, there seems to be no way of knowing which is the correct analysis. In this situation, it appears to be wise not to make a commitment to any particular analysis, but rather to indicate, by the use of some convention, that information belongs in the analysis, but fills an unknown role; and to let the inference mechanism decide. A discussion of "how" questions may be found in [Lehnert, 1978].

v) Garden paths

One of the topics discussed by Marcus is the analysis of garden path sentences, for which he gives a good explanation in terms of the parsing model he has developed [Marcus, 1980]. Among his examples are

THE HORSE RACED PAST THE BARN FELL.
THE BOAT FLOATED DOWN THE RIVER SANK.

These two are examples of a wide range of garden path sentences with one common property: the first verb (RACED or FLOATED) can be seen as either a past participle or as a simple past form; and it is at first perceived as a simple past, and hence as the main verb of the sentence. It is also noteworthy that there is no semantic anomaly in taking the first verb as the main verb; the trouble comes later, when an extraneous word cannot be fitted in to the existing syntactic framework. The correct analysis involves taking these words to be past participles introducing a reduced relative clause.

In AD-HAC, sentences such as these cannot be analysed either. This is because the ATN used to isolate syntactic constituents attempts to find only the right constituents, and for this purpose uses the "PARSING-FAILED" test on arcs to see if any previous analysis path successfully reached a POP state (as mentioned in section 5.2.1); and when a word such as RACED can be analysed as either a participle or as a simple past form, this test ensures that only the simple past is seen. Therefore, sentences such as the examples above cannot be parsed; but if the function PARSING-FAILED is overridden and all analysis paths are taken, those sentences can be analysed.

This seems like an easy solution to the garden-path problem, but unfortunately it is wrong: a sentence such as

"THE CARS MADE IN JAPAN ARE SMALL"

cannot be analysed either, for the same reasons as the genuine garden paths. If this phenomenon is to be handled by AD-HAC, some further work must clearly be undertaken.

Summary

The analyser described here tackles a large variety of phenomena in the domain of single English sentences, and successfully builds representations of their meaning in a formalism which is both far removed from English, and which is hospitable to an inference mechanism. It has its limitations; but the general framework has permitted steady expansion of its coverage of English, and there seems no reason to suppose that its coverage will not be further extended.

Chapter 6: Demonstration of the analyser.

This chapter shows AD-HAC running on three simple texts. The focus here is on the analyser, but the other components are to some extent represented also: the CDforms built by the analyser for the first story are passed over to the sentence generator for paraphrasing, and are then also passed to the inferencer.

The example texts are:

BILL AND JILL WENT TO THE ZOO. THEY GAVE THE MONKEYS SOME PEANUTS, WHICH THEY ATE. THEY WENT TO THE RESTAURANT AND DRANK SOME TEA. JILL TOOK BILL'S MONEY FROM HIM AND GAVE IT TO THE TRAMP WHO WAS TALKING TO THEM. WHAT DID SHE GIVE HIM?

JACK AND JILL WENT UP THE HILL TO FETCH A PAIL OF WATER.
JACK FELL DOWN AND BROKE HIS CROWN, AND JILL CAME TUMBLING AFTER HIM.

A MAN WAS KILLED WHEN THE CAR IN WHICH HE WAS RIDING SWERVED OFF THE ROAD AND STRUCK A TREE.

These texts are entered one sentence at a time, in response to the prompt "Pray continue:". Each input sentence has been placed at the top of a page to facilitate inspection; varying amounts of program output follow.

The following pages show the program's actual output, edited in three ways only: some reformatting has been performed, mostly involving adding and removing blank lines to enhance legibility; some annotation has been added, prefixed with semicolons; and a "parsing graph" for the very first sentence - also produced by the program - has been spliced into the text. The parsing graph appears on page 86, reduced in size to fit a page. Such graphs were produced for the other sentences, but are too large to show here even when photo-reduced.

The analyser normally runs faster than the timing figures shown here would indicate. The gathering of statistics about the performance of the program, and especially the formatted printing of the constituent trees and the conceptual analyses, inflates the time taken by a factor of approximately 2.

6.1) <u>BILL AND JILL GO TO THE ZOO.</u>

This page shows the setting of a few variables whose default values are being overridden for this run of the program, and the function MAIN being called with a few options. These settings account for the difference in the output shown here and that shown in the introduction. The text is the same.

```
@RUN HAC
NIL

*(SETQ syntax T)               ; This setting causes the program to display
                               ; the ATN's component analyses for sentences.
T

*(SETQ plotting T)             ; This causes the analyser to produce graphs
                               ; showing how the analyser's "theories" led
T                              ; to one another, what their preferences were,
                               ; at what level of embedding they operated,
                               ; what class of request was being tried,
                               ; and whether they were tried/untried/forgotten
                               ;
                               ; The first of these graphs has been inserted
                               ; into the material shown here. The others
                               ; are not shown because they are too large.

*(SETQ generation-level 1)     ; This causes the inferencer to invoke the
                               ; generator to express some of its inferences.
1

*(MAIN A I P)                  ; This enters AD-HAC, with the options
                               ;    A to print the analyser's output
                               ;    I to invoke the inferencer.
                               ;    P to paraphrase sentences before
                               ;         inference takes place

                               ; let's go...
```

```
Pray continue:
BILL AND JILL WENT TO THE ZOO.
-----Next sentence    (BILL AND JILL WENT TO THE ZOO)

Component analysis :
(((NP BILL)                                  ; This component analysis is the
    ((CONJ AND)                              ; structure produced by the ATN.
       ((NP JILL)                            ; In this case, each string of
          ((VP(tense PAST)                   ; words has only one syntactic
              (voice ACTIVE)                 ; analysis: 'BILL' is not in the
              (form SIMP)                    ; dictionary as a verb, but it
              (neg NIL)                      ; does have two noun readings.
              (modal NIL)
              (verb GO))
             ((PP (prep TO) (noun ZOO1))
                (((end input))))))))))       ; Marking the end of the sentence

     Analysis is
    theory21                                 ; The name of the winning theory
    ((EVENT (ACTOR (GROUP (1 HUMAN-BILL) (2 JILL)))
            (ACT PTRANS)
            (OBJECT (GROUP (1 HUMAN-BILL) (2 JILL)))
            (FROM DUMMY-PLACE2)
            (TO ZOO1)
            (TIME (NAMED TIMEPOINT2)
                  (COMPARISON (BEFORE *NOW*)))))
    (propernames JILL HUMAN-BILL)            ; Lists provided to give the
    (defrefs ZOO1)                           ; inferencer easy access.

time so far =1.864 seconds (including 0.515 seconds) + 0.000 seconds
                                             ; These figures mean that, in this
                                             ; case, 1.864 seconds were used
                                             ; for the analysis, of which
                                             ; 0.515 seconds were used reading
                                             ; the definition of 'GO' from disc
                                             ; and no time garbage collecting
preference of winning theory is 7
Efficiency of search was 80%                 ; This means that 80% of the
                                             ; theories which were tried lay
                                             ; on the path to the final
                                             ; analysis.
Maximum depth of memory needed 2             ; One (or more) of the theories
                                             ; on the successful analysis path
                                             ; fell to 2nd place in the queue.
Maximum depth of memory searched 2           ; The analyser never evaluated a
                                             ; theory which was placed lower
                                             ; than second in the queue.

Maximum depth of stack built 3
Plotting occupied 0.247 seconds
Total time taken = 2.327 seconds+0.000 seconds
                                             ; Includes gathering and printing
                                             ; all the statistics shown above.
```

```
        theory1
        level 0
        pref=0
        GET
        tried
          |
        theory2
        level 0
        pref=0
        USE
        (BILL)
        tried
       .    |
  theory4   theory3
  level 0   level 0
  pref=-4   pref=2
  EMBED     EMBED
  untried   tried
              |     .
         theory6   theory5
         level 1   level 0
         pref=2    pref=2
         GET       GET
         tried     tried
           |         :
         theory8   theory7
         level 1   level 0
         pref=2    pref=2
         USE       USE
         (JILL)    (AND)
         tried     tried
           |
         theory9
         level 1
         pref=4
         TRIVIAL
         tried
           |
         theory10
         level 1
         pref=5
         UNEMBED
         tried
           |     .
  theory12   theory11
  level 0    level 1
  pref=4     pref=5
  GET        GET
  tried      tried
    |          :
  theory14   theory13
  level 0    level 1
  pref=4     pref=5
  USE        USE
  (WENT)     (WENT)
  tried      tried
    |
  theory15
  level 0
  pref=2
  TRIVIAL
  tried
    |
  theory16
  level 0
  pref=2
  USE-REG
  tried
    |
  theory17
  level 0
  pref=5
  GET
  tried
    |
  theory18
  level 0
  pref=5
  USE
  (TO THE ZOO)
  tried
    |
  theory19
  level 0
  pref=7
  TRIVIAL
  tried
    |
  theory20
  level 0
  pref=7
  GET
  tried
    |
  theory21
  level 0
  pref=7
  USE
  accepted
```

; theory1 has the standard requests for starting.
; It creates one USE theory because there is one
; constituent only.
; theory2 applies the request which seeks a noun
; phrase to start a declarative sentence. The
; presence of "AND" on the constituent tree causes
; the creation of an optional EMBED request, which
; will swallow the word "AND", creating a GROUP
; structure with the present noun phrase and the
; next one.
; Since "BILL" has two noun meanings, a branch
; occurs: the cues associated with proper names,
; and the absence of a determiner, conspire to
; prefer one branch over the other, and theory4
; never comes to the top of the queue and so is
; never tried.
; theory3 has an optional EMBED, so creates two
; new theories: theory5 is not embedded, so when
; "AND" is found in theory7, nothing can be done
; with it and that line of processing is aborted.
; The embedding that occurs in creating theory6
; also swallows AND, and adds a new USE request to
; find another NP. Theory6 gets an NP, and theory7
; happily uses it in the group and adds an UNEMBED
; request. The embedding action in theory3 also
; adds a TRIVIAL request, whose sole effect is to
; increase the preference score by 1: this is done
; in theory9.
; The unembed is optional, so two theories are
; created: the path where unembedding is not done
; comes to grief when it cannot use "WENT". The
; path which did unembed was expecting a verb, and
; so reads in its definition: this is a TRIVIAL
; request (as all verb definitions are), and is
; executed in theory15. Its effect is to build the
; skeleton semantic structure, and to add various
; requests which fill in parts of that structure.
; One of these is the USE-REG which picks up the
; contents of the "subj" register containing the
; (GROUP (1 BILL) (2 JILL)); this is done in
; theory16.
; The next constituent is picked up in theory17,
; and used in theory18 by a PLACE-PP macro.
; The trivial request in theory19 is a remnant of
; a GET-PARTICLE looking for the word "AWAY", and
; fails. theory20 picks up the "(((end input)))"
; which marks the end of the sentence: since all
; constraints are satisifed, the analysis ends at
; theory21.

```
You said : BILL AND JILL WENT TO THE ZOO.
                                        ; Repeating the original sentence
                                        ; (With all these statistics, it
                                        ; is quite possible to forget
                                        ; what you typed!)

My version is : JILL AND BILL WENT TO A ZOO.
                                        ; This paraphrase is produced
                                        ; without reference to the
                                        ; original text: just the
                                        ; conceptual analysis is used.


A certain precondition inference is : JILL AND BILL WERE NEAR THE PLACE FROM
WHERE JILL AND BILL WERE GOING TO A ZOO.
                                        ; The inference mechanism uses
                                        ; the generator to express some
                                        ; of its inferences: because
                                        ; 'generation-level' was set to
                                        ; 1, the inferencer asks for
                                        ; only the more interesting
                                        ; inferences to be expressed.
                                        ; The generator found that it had
                                        ; to indicate "THE PLACE", and
                                        ; so it asked the inferencer for
                                        ; some qualifying information:
                                        ; this is expressed by the
                                        ; relative clause.

A certain resultative inference is : JILL AND BILL ARRIVED AT A ZOO.

An alternative possible motivation inference is : JILL AND BILL DESIRED THAT
JILL AND BILL NOT BE NEAR THE PLACE,FROM WHERE THEY HAD GONE TO A ZOO.
                                        ; There was another "motivation"
                                        ; inference, but the inferencer
                                        ; did not consider it interesting
                                        ; enough to ask for it to be
                                        ; expressed in English. However,
                                        ; it does want this expressed,
                                        ; and, recognising that
                                        ; motivation inferences are often
                                        ; mutually exclusive, it calls
                                        ; this "An alternative motivation"
```

```
Pray continue:
THEY GAVE THE MONKEYS SOME PEANUTS, WHICH THEY ATE.
-----Next sentence   (THEY GAVE THE MONKEYS SOME PEANUTS , WHICH THEY ATE)

Component analysis :
(((NP DUMMY-UNKNOWNS1)
    ((VP(tense PAST)
        (voice ACTIVE)
        (form SIMP)
        (neg NIL)
        (modal NIL)
        (verb GIVE))
       ((NP MONKEYS1)
          ((NP PEANUTS)
             ((isolated-word !,)
                ((RELNP RELPRON1)
                   ((NP DUMMY-UNKNOWNS2)
                      ((VP(tense PAST)
                          (voice ACTIVE)
                          (form SIMP)
                          (neg NIL)
                          (modal NIL)
                          (verb EAT))
                         (((end input)))))))))))

     Analysis is
    theory71
    ((EVENT (ACTOR DUMMY-UNKNOWNS1)
            (ACT ATRANS)
            (OBJECT (FOCUS (EVENT OBJECT))
                    (EVENT (ACTOR DUMMY-UNKNOWNS2)
                           (ACT INGEST)
                           (OBJECT PEANUTS3)
                           (TIME (NAMED TIMEPOINT12)
                                 (COMPARISON (BEFORE *NOW*)))))
            (FROM DUMMY-UNKNOWNS1)
            (TO MONKEYS1)
            (TIME (COMPARISON (AFTER TIMEPOINT2))
                  (NAMED TIMEPOINT9)
                  (COMPARISON (BEFORE *NOW*)))))
    (pronouns DUMMY-UNKNOWNS2 DUMMY-UNKNOWNS1)
    (defrefs MONKEYS1)
    (indefrefs PEANUTS3)

time so far =4.069 seconds (including 0.334 seconds) + 1.686 seconds
preference of winning theory is 6
Efficiency of search was 43%
Maximum depth of memory needed 4
Maximum depth of memory searched 4
Maximum depth of stack built 5
Plotting occupied 0.949 seconds
Total time taken = 5.214 seconds+1.686 seconds
```

You said : THEY GAVE THE MONKEYS SOME PEANUTS,WHICH THEY ATE.

My version is : SOME ENTITIES GAVE THE PEANUTS,WHICH SOME ENTITIES WERE EATING,TO SOME MONKEYS.

; TIMEPOINT9 was "time of giving", and TIMEPOINT12 was "time of eating"
; The analyser could not determine how these times should be related.
; Consequently, the generator finds no information to help its tense
; selection processes, and so uses present progressive, indicating that
; the two times might be the same.
; TIMEPOINT2 was the time of the previous sentence, and the analyser
; assumes that successive sentences in simple past tense relate
; progressively later times.
; At this point, no pronouns have been resolved. The generator uses the
; word 'ENTITIES' to indicate that it doesn't really know what it is
; talking about: this is a deliberate policy, since use of words like
; 'THEM' or 'THEY' can mislead people into thinking that the program
; knows more than it actually does.

A certain precondition inference is : IF IT WAS THE MONKEYS WHICH ATE SOME PEANUTS THEN THE MONKEYS WOULD HAVE THE PEANUTS.

A certain precondition inference is : IF IT WAS JILL AND BILL WHO ATE SOME PEANUTS THEN JILL AND BILL WOULD HAVE THE PEANUTS.

A certain precondition inference is : IF IT WAS JILL AND BILL WHO GAVE SOME PEANUTS TO SOME MONKEYS THEN JILL AND BILL WOULD HAVE THE PEANUTS.

A certain resultative inference is : IF IT WAS JILL AND BILL WHO GAVE SOME PEANUTS TO SOME MONKEYS THEN THE MONKEYS WOULD ACQUIRE THE PEANUTS.

A possible follevent inference is : SOME MONKEYS STARTED EATING SOME PEANUTS.

Pronoun resolution : IT WAS JILL AND BILL WHO GAVE SOME PEANUTS TO SOME MONKEYS.

Pronoun resolution : IT WAS THE MONKEYS WHICH ATE SOME PEANUTS.

```
Pray continue:
THEY WENT TO THE RESTAURANT AND DRANK SOME TEA.
-----Next sentence    (THEY WENT TO THE RESTAURANT AND DRANK SOME TEA)

Component analysis :
(((NP DUMMY-UNKNOWNS3)
    ((VP(tense PAST)
        (voice ACTIVE)
        (form SIMP)
        (neg NIL)
        (modal NIL)
        (verb GO))
       ((PP (prep TO) (noun RESTAURANT1))
          ((CONJ AND)                  ; Note that the ATN's analysis does not
             ((VP(tense PAST)          ; distinguish in any way the role
                 (voice ACTIVE)        ; of this conjunction: the analysis is
                 (form SIMP)           ; the same as for the conjunction in
                 (neg NIL)             ; the first sentence, which served
                 (modal NIL)           ; there to conjoin nominals. Here, it
                 (verb DRINK))         ; conjoins clauses.
                ((NP TEA)
                   (((end input)))))))))


     Analysis is
    theory31
    ((CONJUNCT (FIRST (EVENT (ACTOR DUMMY-UNKNOWNS3)
                             (ACT PTRANS)
                             (OBJECT DUMMY-UNKNOWNS3)
                             (FROM DUMMY-PLACE4)
                             (TO RESTAURANT1)
                             (TIME (COMPARISON (AFTER TIMEPOINT9))
                                   (NAMED TIMEPOINT41)
                                   (COMPARISON (BEFORE *NOW*)))))
               (SECOND (EVENT (ACTOR DUMMY-UNKNOWNS3)    ; note that the same
                              (ACT INGEST)               ; subject is used
                              (OBJECT TEA1)              ; for both clauses.
                              (TIME (COMPARISON (AFTER TIMEPOINT9))
                                    (NAMED TIMEPOINT44)
                                    (COMPARISON (BEFORE *NOW*))
                                    (COMPARISON (AFTER TIMEPOINT41))))))
    (pronouns DUMMY-UNKNOWNS3)
    (defrefs RESTAURANT1)
    (indefrefs TEA1)

time so far =2.089 seconds (including 0.413 seconds) + 0.000 seconds
preference of winning theory is 10
Efficiency of search was 80%
Maximum depth of memory needed 2
Maximum depth of memory searched 2
Maximum depth of stack built 3
Plotting occupied 0.333 seconds
Total time taken = 2.636 seconds+0.000 seconds
```

You said : THEY WENT TO THE RESTAURANT AND DRANK SOME TEA.

My version is : SOME ENTITIES MOVED TO A RESTAURANT AND THOSE ENTITIES DRANK SOME TEA.

A certain precondition inference is : THE ENTITIES,WHICH WERE MOVING TO A RESTAURANT,WERE NEAR SOMEWHERE,OTHER THAN THE RESTAURANT.

A certain resultative inference is : THE ENTITIES,WHICH HAD MOVED TO A RESTAURANT,ARRIVED AT THE RESTAURANT.

An alternative possible motivation inference is : IF IT WAS JILL AND BILL WHO MOVED TO A RESTAURANT THEN JILL AND BILL WOULD DESIRE THAT THEY NOT BE NEAR THE PLACE,WHERE THEY HAD BEEN.

A certain precondition inference is : IF IT WAS JILL AND BILL WHO MOVED TO A RESTAURANT THEN JILL AND BILL WOULD HAVE SOME TEA.

An alternative possible motivation inference is : IF IT WAS THE MONKEYS WHICH MOVED TO A RESTAURANT THEN THE MONKEYS WOULD DESIRE THAT THEY NOT BE NEAR THE PLACE,WHERE THEY HAD BEEN.

A certain precondition inference is : IF IT WAS THE MONKEYS WHICH MOVED TO A RESTAURANT THEN THE MONKEYS WOULD HAVE SOME TEA.

Pronoun resolution : IT WAS JILL AND BILL WHO MOVED TO A RESTAURANT.

```
Pray continue:
JILL TOOK BILL'S MONEY FROM HIM AND GAVE IT TO THE TRAMP WHO WAS TALKING TO
THEM.
-----Next sentence   (JILL TOOK BILL'S MONEY FROM HIM AND GAVE IT TO THE
                      TRAMP WHO WAS TALKING TO THEM)

Component analysis :
(((NP JILL)
    ((VP(tense PAST)
        (voice ACTIVE)
        (form SIMP)
        (neg NIL)
        (modal NIL)
        (verb TAKE))
       ((NP(FOCUS (STATE THING))
           (STATE (STATENAME POSS) (THING MONEY) (VAL BILL)))
          ((PP (prep FROM) (noun DUMMY-MALE1))
             ((CONJ AND)
                ((VP(tense PAST)
                    (voice ACTIVE)
                    (form SIMP)
                    (neg NIL)
                    (modal NIL)
                    (verb GIVE))
                   ((NP DUMMY-THING1)
                      ((PP (prep TO) (noun TRAMP1))
                         ((RELNP RELPRON1)
                            ((VP(tense PAST)
                                (voice ACTIVE)
                                (form PROG)
                                (neg NIL)
                                (modal NIL)
                                (verb TALK))
                               ((PP (prep TO) (noun DUMMY-UNKNOWNS4))
                                  (((end input))))))
                         ((WHNP ?DUMMY-HUMAN4)
                            ((VP(tense PAST)
                                (voice ACTIVE)
                                (form PROG)
                                (neg NIL)
                                (modal NIL)
                                (verb TALK))
                               ((PP (prep TO) (noun DUMMY-UNKNOWNS4))
                                  (((end input)))))))))))))
```

```
 Analysis is
theory77
((CONJUNCT (FIRST (EVENT (ACTOR JILL)
                         (ACT ATRANS)
                         (OBJECT (FOCUS (STATE THING))
                                 (STATE (STATENAME POSS)
                                        (THING MONEY3)
                                        (VAL HUMAN-BILL)
                                        (TIME (NAMED TIMEPOINT72)
                                              (COMPARISON
                                                 (BEFORE *NOW*)))))
                         (TO JILL)
                         (FROM DUMMY-MALE1)
                         (TIME (COMPARISON (AFTER TIMEPOINT44))
                               (NAMED TIMEPOINT72)
                               (COMPARISON (BEFORE *NOW*)))))
           (SECOND (EVENT (ACTOR JILL)
                          (ACT ATRANS)
                          (OBJECT DUMMY-THING1)
                          (FROM JILL)
                          (TO (FOCUS (EVENT ACTOR))
                              (EVENT (ACTOR TRAMP1)
                                     (ACT MTRANS)
                                     (MOBJECT CONCEPTS1)
                                     (FROMCP TRAMP1)
                                     (TOCP DUMMY-UNKNOWNS4)
                                     (TIME (NAMED TIMEPOINT75)
                                           (COMPARISON (BEFORE *NOW*))
                                           (COMPARISON
                                              (AFTER TIMEPOINT72)))))
                          (TIME (COMPARISON (AFTER TIMEPOINT44))
                                (NAMED TIMEPOINT75)
                                (COMPARISON (BEFORE *NOW*))
                                (COMPARISON (AFTER TIMEPOINT72)))))))
(pronouns DUMMY-UNKNOWNS4 DUMMY-THING1 DUMMY-MALE1)
(propernames HUMAN-BILL JILL)
(defrefs TRAMP1)
(indefrefs MONEY3)

time so far =6.375 seconds (including 0.687 seconds) + 1.568 seconds
preference of winning theory is 17
Efficiency of search was 60%
Maximum depth of memory needed 3
Maximum depth of memory searched 5
Number of theories forgotten 1
Plotting occupied 1.128 seconds
Total time taken = 7.766 seconds+1.568 seconds
```

You said : JILL TOOK BILL'S MONEY FROM HIM AND GAVE IT TO THE TRAMP WHO WAS TALKING TO THEM.

My version is : JILL TOOK BILL'S MONEY AND JILL GAVE SOMETHING TO THE TRAMP, WHO WAS STATING SOMETHING.

A certain precondition inference is : THE ENTITIES,TO WHICH A TRAMP HAD SAID SOMETHING,WERE NEAR THE TRAMP.

A probable motivation inference is : A TRAMP DESIRED THAT THE ENTITIES,TO WHICH THE TRAMP HAD SAID SOMETHING,PONDER.

A probable instrumental inference is : A TRAMP SPOKE.

A probable resultative inference is : IF THE ENTITIES,TO WHICH A TRAMP WAS SAYING SOMETHING,LISTENED TO THE TRAMP THEN THOSE ENTITIES WOULD PONDER.

A possible normative inference is : IF THE ENTITIES,TO WHICH A TRAMP WAS SAYING SOMETHING,DIDN'T LISTEN TO THE TRAMP THEN HE WOULD START TO BE ANGRY WITH THOSE ENTITIES.

A certain precondition inference is : THE MALE,FROM WHOM JILL WAS TAKING SOME MONEY,HAD THE MONEY.

A certain resultative inference is : JILL ACQUIRED SOME MONEY.

A certain motivation inference is : JILL WANTED SOME MONEY.

A probable resultative inference is : IF THE MALE,WHO HAD SOME MONEY,DIDN'T GIVE JILL PERMISSION TO TAKE THE MONEY THEN JILL WOULD BE REMORSEFUL.

A probable normative inference is : IF THE MALE,FROM WHOM JILL HAD TAKEN SOME MONEY,THOUGHT THAT THE PERSON WHO WAS TAKING THE MONEY HAD TAKEN IT THEN THAT MALE WOULD TELL THE POLICE THAT THAT PERSON TOOK IT.

A certain normative inference is : IF THE MALE,FROM WHOM JILL HAD TAKEN SOME MONEY,THOUGHT THAT THE PERSON WHO WAS TAKING THE MONEY HAD TAKEN IT THEN THAT MALE WOULD WANT IT.

A probable resultative inference is : IF THE MALE,FROM WHOM JILL HAD TAKEN SOME MONEY,THOUGHT THAT IT BE JILL WHO TOOK THE MONEY THEN THAT MALE WOULD HOPE TO TAKE IT FROM JILL.

A certain normative inference is : IF THE MALE,FROM WHOM JILL HAD TAKEN SOME MONEY,DIDN'T THINK THAT IT BE JILL TOOK THE MONEY THEN THAT MALE WOULD START TO DESIRE THAT HE KNOW.

A certain precondition inference is : JILL HAD THE THING WHICH JILL WAS GIVING TO A TRAMP.

A certain resultative inference is : A TRAMP ACQUIRED THE THING WHICH JILL WAS GIVING TO THE TRAMP.

Pronoun resolved : IT WAS BILL FROM WHOM JILL TOOK SOME MONEY.

Pronoun resolved : IT WAS THE MONEY WHICH JILL GAVE TO A TRAMP.

Pronoun resolved : IT WAS JILL AND BILL TO WHOM A TRAMP SAID SOMETHING.

```
Pray continue:
WHAT DID SHE GIVE HIM?
-----Next sentence   (WHAT DID SHE GIVE HIM)

Component analysis :
(((RELNP RELPRON1)
    ((VP(tense PAST)
        (voice ACTIVE)
        (form SIMP)
        (neg NIL)
        (modal NIL)
        (verb DO))
       ((NP DUMMY-FEMALE1)
          ((VP(tense PRES)
              (voice ACTIVE)
              (form UNTENSED)
              (neg NIL)
              (modal NIL)
              (verb GIVE))
             ((NP DUMMY-MALE2)
                (((end input)))))))
   ((WHNP ?DUMMY-THING4)
      ((VP(tense PAST)
          (voice ACTIVE)
          (form SIMP)
          (neg NIL)
          (modal NIL)
          (verb DO))
         ((NP DUMMY-FEMALE1)
            ((VP(tense PRES)
                (voice ACTIVE)
                (form UNTENSED)
                (neg NIL)
                (modal NIL)
                (verb GIVE))
               ((NP DUMMY-MALE2)
                  (((end input))))))))
```

```
 Analysis is
theory40
((FOCUS (EVENT OBJECT))
 (EVENT (ACTOR DUMMY-FEMALE1)
        (ACT ATRANS)
        (OBJECT ?DUMMY-THING4)
        (FROM DUMMY-FEMALE1)
        (TO DUMMY-MALE2)
        (TIME (NAMED TIMEPOINT145)
              (COMPARISON (BEFORE *NOW*)))))
(pronouns DUMMY-MALE2 DUMMY-FEMALE1)
(queries ?DUMMY-THING4)
```

```
time so far =3.519 seconds (including 0.278 seconds) + 1.654 seconds
preference of winning theory is 6
Efficiency of search was 52%
Maximum depth of memory needed 2
Maximum depth of memory searched 2
Number of theories forgotten 1
Plotting occupied 2.170 seconds
Total time taken = 5.918 seconds+3.334 seconds
 You said : WHAT DID SHE GIVE HIM.


My version is : WHAT DID A FEMALE GIVE TO A MALE?


A certain precondition inference is : THE FEMALE,WHO WAS GIVING SOMETHING TO
A MALE,HAD THAT THING.


A certain resultative inference is : THE MALE,WHO WAS RECEIVING
SOMETHING,ACQUIRED THAT THING.


Pronoun resolved : IT WAS JILL WHO GAVE SOMETHING TO A MALE.


Answering question (1): JILL GAVE SOME MONEY TO A TRAMP.


Pray continue:
(Returning to Lisp...
```

Note: The inference processes will be demonstrated in Chapter 9, using this same example, and taking for granted the behaviour of the analyser and of the generator.

6.2) <u>JACK AND JILL WENT UP THE HILL TO FETCH A PAIL OF WATER.</u>

This well-known story has to be modified for AD-HAC to handle it; the traditional second line is

JACK FELL DOWN AND BROKE HIS CROWN, AND JILL CAME TUMBLING AFTER.

Unfortunately, AD-HAC's analyser cannot handle this as it stands: it expects something to follow the word "AFTER". Further work would enable the analyser to handle the unedited text. Meanwhile, the simplest modification is to add the word "HIM", which is the approach I have adopted here.

There are however some tricky points in this text, which AD-HAC deals with quite successfully. The use of the purposive TO-complement in the first sentence, the two ways of using the word "AND", and the expression "CAME TUMBLING", are all successfully handled.

The initial variable settings are not shown for this example. The CDforms generated for these two sentences have been reformatted in order to fit them onto these pages, and multiple specifications of time relationships have in some cases been eliminated for the same reason. It must be emphasised however that no substantive alterations have been made.

```
Pray continue:
JACK AND JILL WENT UP THE HILL TO FETCH A PAIL OF WATER.
-----Next sentence    (JACK AND JILL WENT UP THE HILL TO FETCH A PAIL OF
                       WATER)

Component analysis :
(((NP JACK)
    ((CONJ AND)
       ((NP JILL)
          ((VP(tense PAST)
              (voice ACTIVE)
              (form SIMP)
              (neg NIL)
              (modal NIL)
              (verb GO))
             ((PP (prep UP) (noun HILL1))
                ((VP(tense PRES)
                    (voice ACTIVE)
                    (form INF)
                    (neg NIL)
                    (modal NIL)
                    (verb FETCH))
                   ((NP PAIL)
                      ((PP (prep OF) (noun WATER))
                         (((end input)))))))))))
```

```
 Analysis is
theory54
((CAUSE
   (ANTECEDENT
       (STATE
          (STATENAME MLOC)
          (INCP (GROUP (1 JACK) (2 JILL)))
          (TIME (NAMED TIMEPOINT2)
                (COMPARISON (BEFORE *NOW*)))
          (MOBJECT
              (CAUSE
                  (ANTECEDENT
                      (EVENT (ACTOR (GROUP (1 JACK) (2 JILL)))
                             (ACT PTRANS)
                             (OBJECT (GROUP (1 JACK) (2 JILL)))
                             (FROM DUMMY-PLACE2)
                             (TO (FOCUS (STATE THING))                       ;
                                 (STATE (STATENAME PART)                     ;  see
                                        (THING *HIGHPART*1)                  ; below
                                        (VAL HILL1)                          ;
                                        (TIME (NAMED TIMEPOINT2))))          ;
                             (TIME (NAMED TIMEPOINT2))))
                  (RESULT
                      (EVENT (ACTOR (GROUP (1 JACK) (2 JILL)))
                             (ACT PTRANS)
                             (OBJECT
                                (GROUP
                                   (1 (GROUP (1 JACK) (2 JILL)))
                                   (2 (FOCUS (STATE THING))
                                      (STATE (STATENAME QUANTITY)
                                             (THING WATER1)
                                             (VAL *PAILFUL*1)
                                             (TIME (NAMED TIMEPOINT3))))))
                             (TO DUMMY-PLACE3)
                             (FROM DUMMY-PLACE4)
                             (TIME (NAMED TIMEPOINT3)
                                   (COMPARISON
                                       (AFTER
                                           (NAMED TIMEPOINT2))))
                             (ABILITY CAN)))))))
   (RESULT
       (EVENT (ACTOR (GROUP (1 JACK) (2 JILL)))
              (ACT PTRANS)
              (OBJECT (GROUP (1 JACK) (2 JILL)))
              (FROM DUMMY-PLACE2)
              (TO *HIGHPART*1)
              (TIME (NAMED TIMEPOINT2)
                    (COMPARISON (BEFORE *NOW*)))))))
 (propernames JILL JACK)
 (defrefs HILL1)
 (indefrefs WATER1 *PAILFUL*1)

; The underlining of *HIGHPART*1 indicates that the token shown is
; embedded in a modifying structure which has already been shown.
```

```
time so far =5.698 seconds (including 0.794 seconds) + 0.000 seconds
preference of winning theory is 4
Efficiency of search was 63%
Maximum depth of memory needed 3
Maximum depth of memory searched 3
Maximum depth of stack built 5
Total time taken = 5.939 seconds+0.000 seconds

 You said : JACK AND JILL WENT UP THE HILL TO FETCH A PAIL OF WATER.

My version is : JILL AND JACK WENT TO A HILL'S UPPER REACHES IN ORDER TO
CONVEY THE *PAILFUL*1 WATER.

     ; The generator has placed a bit of CD structure into the sentence!
     ; This has occurred because the state QUANTITY is not known to the
     ; generator: when the generator encounters unknown states, it treats
     ; them as communicating some information which should be expressed as
     ; an adjective. Clearly, the generator should be told about QUANTITY.
```

Pray continue:
JACK FELL DOWN AND BROKE HIS CROWN, AND JILL CAME TUMBLING AFTER HIM.
-----Next sentence (JACK FELL DOWN AND BROKE HIS CROWN , AND JILL CAME TUMBLING AFTER HIM)

Component analysis :

```
(((NP JACK)
    ((VP(tense PAST)
        (voice ACTIVE)
        (form SIMP)
        (neg NIL)
        (modal NIL)
        (verb FALL))
       ((isolated-word DOWN)
          ((CONJ AND)
             ((VP(tense PAST)
                 (voice ACTIVE)
                 (form SIMP)
                 (neg NIL)
                 (modal NIL)
                 (verb BREAK))
                ((NP(FOCUS (STATE THING))
                    (STATE (STATENAME POSS)
                           (THING CROWN)
                           (VAL DUMMY-MALE1)))
                   ((isolated-word !,)
                      ((CONJ AND)
                         ((NP JILL)
                            ((VP(tense PAST)
                                (voice ACTIVE)
                                (form SIMP)
                                (neg NIL)
                                (modal NIL)
                                (verb COME))
                               ((NP TUMBLING)
                                  ((CONJ AFTER)
                                     ((NP DUMMY-MALE2)
                                        (((end input))))))
                               ((VP(tense PRES)
                                   (voice ACTIVE)
                                   (form PARTICIPLE)
                                   (neg NIL)
                                   (modal NIL)
                                   (verb TUMBLE))
                                  ((CONJ AFTER)
                                     ((NP DUMMY-MALE2)
                                        (((end input))))))))))))))))
```

```
 Analysis is
theory88
((CONJUNCT
   (FIRST
      (EVENT (ACTOR *GRAVITY*)
             (ACT PTRANS)
             (OBJECT JACK)
             (FROM DUMMY-PLACE5)
             (TO *GROUND*)
             (TIME (NAMED TIMEPOINT4)
                   (COMPARISON (BEFORE *NOW*)))))
   (SECOND
      (CONJUNCT
         (FIRST
            (CAUSE
               (ANTECEDENT
                   (EVENT (ACTOR JACK)
                          (ACT DO)
                          (TIME (NAMED TIMEPOINT6)
                                (COMPARISON (BEFORE *NOW*))
                                (COMPARISON (AFTER TIMEPOINT4)))))
               (RESULT
                   (STATE (STATENAME PSTATE)
                          (THING (FOCUS (STATE THING))
                                 (STATE (STATENAME POSS)
                                        (THING PATE2)
                                        (VAL DUMMY-MALE1)
                                        (TIME (NAMED TIMEPOINT6))))
                          (VAL (LOWERBY 4))
                          (TIME (NAMED TIMEPOINT6)
                                (COMPARISON (BEFORE *NOW*))
                                (COMPARISON (AFTER TIMEPOINT4)))))))
         (SECOND (EVENT (ACTOR *GRAVITY*)
                        (ACT PTRANS)
                        (OBJECT JILL)
                        (FROM DUMMY-PLACE8)
                        (TO *GROUND*)
                        (TIME (COMPARISON
                                  (AFTER
                                      (FOCUS (EVENT TIME))
                                      (EVENT (ACTOR *GRAVITY*)
                                             (ACT PTRANS)
                                             (OBJECT DUMMY-MALE2)
                                             (FROM DUMMY-PLACE8)
                                             (TO *GROUND*)
                                             (TIME (NAMED TIMEPOINT9)))))
                              (NAMED TIMEPOINT7)
                              (COMPARISON (BEFORE *NOW*))
                              (COMPARISON (AFTER TIMEPOINT4))
                              (COMPARISON (AFTER TIMEPOINT6))))))))
   (pronouns DUMMY-MALE2 DUMMY-MALE1)
   (propernames JILL JACK)
   (indefrefs PATE2)
```

time so far =8.390 seconds (including 0.916 seconds) + 1.569 seconds
preference of winning theory is 19
Efficiency of search was 58%
Maximum depth of memory needed 4
Maximum depth of memory searched 4
Number of theories forgotten 1
Total time taken = 8.625 seconds+1.569 seconds

You said : JACK FELL DOWN AND BROKE HIS CROWN,AND JILL CAME TUMBLING AFTER HIM.

My version is : JACK FELL DOWN AND JACK DAMAGED A MALE'S PATE AND JILL FELL DOWN AFTER A MALE FELL DOWN.

Pray continue:
(Returning to Lisp...

Note: The inferencer to be described in Chapter 8 has not yet been equipped with sufficient inference networks to handle this simple story. In particular, no inferences are yet drawn from the state MLOC. Since the representation of the first sentence has an MLOC as one of its principal elements, the inferencer's performance on this story is not particularly impressive.

6.3) A MAN WAS KILLED WHEN ......

The final example demonstrates the analyser's strengths and weaknesses. [Cullingford, 1978] quotes the sentence

A NEW JERSEY MAN WAS KILLED FRIDAY EVENING WHEN THE CAR IN WHICH HE WAS RIDING SWERVED OFF ROUTE 69 AND STRUCK A TREE.

noting that ELI (the later version of Riesbeck's analyser) "was not up to analysing the complicated, nested relative clauses this sentence contains".

AD-HAC is quite capable of handling the complex nested relative clauses, but cannot handle the phrases

- A NEW JERSEY MAN
- FRIDAY EVENING
- ROUTE 69

The remaining pages of this chapter show the analysis built by AD-HAC for a simplified version. The conceptual structure has again been reformatted, and redundant time specifications have been removed.

Pray continue:
A MAN WAS KILLED WHEN THE CAR IN WHICH HE WAS RIDING SWERVED OFF THE ROAD AND STRUCK A TREE.

```
-----Next sentence   (A MAN WAS KILLED WHEN THE CAR IN WHICH HE WAS RIDING
                      SWERVED OFF THE ROAD AND STRUCK A TREE)
Component analysis :
(((NP MAN)
    ((VP(tense PAST)
        (voice PASS)
        (form SIMP)
        (neg NIL)
        (modal NIL)
        (verb KILL))
       ((CONJ WHEN)
          ((NP CAR1)
             ((RELPP (prep IN) (noun RELPRON1))
                ((NP DUMMY-MALE1)
                   ((VP(tense PAST)
                       (voice ACTIVE)
                       (form PROG)
                       (neg NIL)
                       (modal NIL)
                       (verb RIDE))
                      ((VP(tense PAST)
                          (voice ACTIVE)
                          (form SIMP)
                          (neg NIL)
                          (modal NIL)
                          (verb SWERVE))
                         ((PP (prep OFF) (noun ROAD1))
                            ((CONJ AND)
                               ((VP(tense PAST)
                                   (voice ACTIVE)
                                   (form SIMP)
                                   (neg NIL)
                                   (modal NIL)
                                   (verb STRIKE))
                                  ((NP TREE)
                                     (((end input))))))))))))))
```

```
 Analysis is
theory83
((CAUSE
   (ANTECEDENT
     (EVENT (ACTOR DUMMY-HUMAN1)
            (ACT DO)
            (TIME (NAMED TIMEPOINT1)
                  (COMPARISON (BEFORE *NOW*))))))
   (RESULT
     (STATE (STATENAME HEALTH)
            (THING MAN1)
            (VAL -10)
            (TIME (NAMED TIMESPAN1)
                  (COMPARISON (BEFORE *NOW*))
                  (TS (COMPARISON
                        (NOTBEFORE
                           (FOCUS (CONJUNCT FIRST EVENT TIME))
                           (CONJUNCT
                              (FIRST
                                 (EVENT (ACTOR *MOMENTUM*)
                                        (ACT PTRANS)
                                        (OBJECT
                                           (FOCUS (EVENT ACTOR))
                                           (EVENT (ACTOR CAR1)
                                                  (ACT PTRANS)
                                                  (OBJECT
                                                     (GROUP (1 CAR1)
                                                            (2 DUMMY-MALE1)))
                                                  (FROM DUMMY-PLACE2)
                                                  (TO DUMMY-PLACE1)
                                                  (TIME (NAMED TIMESPAN3)
                                                        (COMPARISON
                                                           (BEFORE *NOW*)))))
                                        (FROM ROAD1)
                                        (TO DUMMY-PLACE6)
                                        (TIME (NAMED TIMEPOINT10))))
                              (SECOND
                                 (EVENT (ACTOR *MOMENTUM*)
                                        (ACT PROPEL)
                                        (OBJECT CAR1)
                                        (FROM DUMMY-PLACE7)
                                        (TO TREE1)
                                        (TIME (NAMED TIMEPOINT14)
                                              (COMPARISON
                                                 (AFTER TIMEPOINT10)))))))
                        (NAMED TIMEPOINT1)
                        (COMPARISON (BEFORE *NOW*))))))))
    (pronouns DUMMY-MALE1)
    (defrefs ROAD1 CAR1)
    (indefrefs TREE1 MAN1)
```

```
time so far =7.764 seconds (including 1.187 seconds) + 0.000 seconds
preference of winning theory is 0
Efficiency of search was 67%
Maximum depth of memory needed 3
Maximum depth of memory searched 3
Maximum depth of stack built 5
Total time taken = 8.059 seconds+0.000 seconds
Pray continue:
(Returning to Lisp...
```

Chapter 7: Inference 1: Background

Story understanding requires the application of real-world knowledge, which often takes the form of plausible inferences as distinct from logical inferences. Consequently, although Predicate Calculus is a proven formalism in which deductive rules may be easily stated and applied, the non-deductive nature of the inference rules required for understanding texts has led many researchers to adopt different models of inference; though Hayes has argued that logic has been dismissed for questionable reasons [Hayes1, 1977].

The inference mechanism used in AD-HAC is described in Chapter 8. It represents a return to the style of inference adopted by Rieger in his MEMORY program, and earlier by Charniak, but with critical modifications designed to overcome the defects of these earlier systems. As elaborated below, Rieger's program suffered from problems of combinatorial explosion which left it unable to handle stories about stereotyped situations. These problems prompted Schank and Abelson to postulate higher-level knowledge structures, "Scripts" and "Plans", and to see the story-understanding process as largely the matching of story statements onto these pre-stored structures. The approach adopted in AD-HAC is rather different: where scripts and plans emphasise higher-level knowledge, AD-HAC's inference processes are driven mostly by low-level knowledge structures which have the ability to invoke script-like and plan-like structures when appropriate.

In this chapter, I trace the development of these ideas by describing the work of Charniak and Rieger, the experimental systems SAM and PAM (which exploited scripts and plans respectively), and Rieger's later work on conceptual overlays. For further details, the reader is referred to [Charniak, 1972], [Rieger, 1974], [Schank and Abelson, 1977], [Wilensky, 1977], [Cullingford, 1978], and [Rieger, 1975].

With the exception of Charniak's work, the systems described here are based on CD; but this is not the justification for describing only these systems. Inference is one of the central topics in artificial intelligence research, and models of inference have taken many forms. It is necessary therefore to restrict attention here to those earlier systems which are particularly relevant to AD-HAC. As the next chapter shows, AD-HAC exploits some of the key ideas which characterised the work discussed here: inference processes associated with conceptual primitives; the identification of classes of inference; the formulation of inference processes as networks; the use of mundane knowledge; the use of higher-level (script-like) knowledge; and the use of "certainty ratings" to permit the use of uncertain information.

7.1) Charniak: Deep Semantic Processing

Charniak's work was concerned with the representation of common-sense knowledge, and particularly with how such knowledge could be brought to bear in the understanding of children's stories. [Charniak, 1972] describes several aspects of a theoretical model, and a partial implementation of this model in an unnamed computer program, which, for convenience, I shall call DSP (for "Deep Semantic Processing").

Neither DSP, nor the theoretical model, was particularly concerned with the processes of sentence analysis or sentence generation, but were assumed to take as input a representation of the meaning of sentences (broadly speaking, as sets of elementary propositions), and to return similar representations as output. The inputs were intended to reflect only that information which could be extracted from a sentence in isolation: the model, and the program DSP, then supplied and applied knowledge of the real world, resulting in an understanding of these sentences in context.

DSP used Micro-Planner (henceforward "uPlanner"), a language designed for theorem-proving, based on pattern-matching and providing automatic backtracking. Charniak's theoretical model is in fact couched in terms which assume the availability of these features, and so corresponds in many details to DSP, the partial implementation: I shall therefore speak of the program and the theory as though they were the same, and I shall use the term DSP to cover them both.

7.1.1) DSP's basic machinery

uPlanner maintains a data base of assertions, and provides facilities for storing, retrieving and erasing these assertions. uPlanner programs typically work by stating goals to be achieved; theorems provide methods of 'proving' goals, often by setting up subgoals; the uPlanner system then tries to satisfy the stated goals either by finding suitable assertions in the data base, or by setting up and proving appropriate subgoals. Brief descriptions may be found in [Charniak, 1972] and [Winograd, 1971]; it is also discussed by [Sussman and Winograd, 1971] and [Bobrow and Raphael, 1974].

In DSP, the input representations were converted into uPlanner assertions for storage and manipulation, while the world knowledge was held as uPlanner theorems. In particular, uPlanner's antecedent theorems were heavily used. An antecedent theorem has several parts, of which the most important are the pattern and the body. In a uPlanner program, whenever an assertion is added to the data base, the stock of antecedent theorems is scanned, and any theorem whose pattern can be matched with the new assertion is fired, ie. its body is executed. In practical terms, this means that an antecedent theorem will lie in wait until a certain condition arises, and will then spring into action and cause some program fragment - the theorem's body - to be executed. Such program fragments may be called demons.

DSP had three principal components, called demons, base routines and bookkeeping, and also subsidiary fact finders. In Charniak's own words,

Demons	- Facts which are introduced by "concepts" occurring in the story are called "demons" since in many cases they must wait for further information. In such cases we can think of them "looking" for the appropriate fact. So "not being willing to trade" might put in a demon looking for a better offer.
Base routines	- These constitute what we know about a "concept" independent of "context". So, for example, if A gives B to C then C now "has" B. This is not dependent on what happened earlier in the story.
Bookkeeping	- This does chores like keeping the data base relatively consistent and non-redundant. So, should a person in the story change location, we must update the old location statement.

Fact finders - These are utility routines for doing standard deductions which aren't worth asserting separately. A typical fact finder might say, "If you want to know if person P knows fact F, just see if when F occurred, (or was said by some character in the story) P was around."

Charniak argued that to have all DSP's knowledge always accessible would have deleterious effects. Firstly, the program would be slower and would occupy more memory space. Secondly, and much more importantly, irrelevant knowledge could be mistakenly applied and would result in an incorrect understanding of the story. (Indeed, special steps were taken in DSP to remove potentially relevant demons which have outlived their usefulness, since these might otherwise be mistakenly applied later on and cause similar problems.) Charniak therefore introduced the notion of "concept" as a handle on a body of knowledge potentially relevant to inputs invoking that concept.

DSP works by keeping assertions both in the data base, and on a list of assertions "TO-BE-DONE"; to each assertion on this list, the three main components described above are applied, and new assertions are placed at the end of this list and processed in their turn. In discussing the order of application of these three components, Charniak concludes that base routines must be split into two sections, one corresponding to knowledge about simple entailments which can be stated without knowing the referents of noun phrases, and one corresponding to more complex knowledge - typically manifested as the introduction of demons - which does depend on knowing these referents. The application of these components is then shown to be

Demons
Base routines, part 1
Bookkeeping
Base routines, part 2

7.1.2) DSP's machinery in action

The sentences comprising a story were translated by hand into the input format, which was closely related to uPlanner's assertion format, but in which noun phrases particularly required further processing in order to determine reference. DSP's reference determination is covered in the next section: briefly, the conversion into the assertion format requires that uPlanner tokens are created to represent these noun phrases, are associated with relevant descriptive information, and are substituted in the appropriate places.

Thus, the sentence "Janet baked a cake" might come to be represented as a set of assertions, which can be given in a simplified form as follows:

```
(BAKE JANET1 CAKE1)
(IS JANET1 PERSON)
(NAME JANET1 JANET)
(SEX JANET1 FEMALE)
(AGE JANET1 YOUNG)
(IS CAKE1 CAKE)
```

(DSP's actual assertions are somewhat more complicated, as they include assertion numbers, tense markers, negation markers, assertion types and property lists. These details are omitted for the present illustrative purpose.)

As each assertion is made, any preexisting demon whose pattern matches the assertion will fire. Such demons may have been created in response to earlier lines of the story. When the process of adding assertions is completed, the appropriate (first part) base routines go to work. These base routines are associated with concepts: predicates - BAKE,IS,NAME etc. - and classes of object, eg. CAKE. The base routines each function as a handle on knowledge specific to a concept: they therefore add new assertions corresponding to entailments, and add new demons to pick up incoming assertions and give them a context-dependent interpretation.

Whilst DSP depends crucially upon the ability of demons to anticipate further information, Charniak demonstrates that in general demons must also be allowed to apply to existing information. For example, the concept 'RAIN' might add a demon embodying the knowledge that any person who is outside while it is raining will get wet. This demon will have a pattern, say "(OUTSIDE ?PERSON)" (*1), and the body will add a new assertion, say "(BECOME-WET ?PERSON)". In a text such as

"It was raining. Fred was outside."

this demon is added by the concept 'RAIN' introduced by the first sentence. When the second sentence is converted to assertions, the demon fires, matching ?PERSON with FRED1 (say) and asserting that Fred becomes wet. However, in a trivially different text,

"Fred was outside. It was raining."

the straightforward demon-application mechanisms of uPlanner would fail to draw this conclusion. This is a pervasive problem, so DSP provides an alternative method of demon introduction which permits demons to look back at old assertions, as well as waiting for new ones. Without such an ability, knowledge would have to be duplicated and scattered throughout the system: in the example above, the concept 'OUTSIDE' would have to be equipped with a demon which waits for a RAIN assertion, and then asserts that the person outside will get wet. Naturally, Charniak sought to avoid such duplication.

However, there is a further case where, if the system is to perform acceptably, special steps must also be taken to avoid the repetition of information. Demons are largely used to relate new information to old: thus, to handle the text

"Janet was going to get a present for Jack. She needed some money"

the 'GET PRESENT' concept will introduce a demon which looks for an assertion that the present-getter needs money, and responds by asserting that the need for money is a result of (intending to) get a present. Similarly, for

"Janet needed some money. She went to get her piggybank"

the 'GET PIGGYBANK' concept introduces a demon which looks for a need-money assertion, and responds by asserting that the need for money is the reason for fetching the piggybank. However, for the text

"Janet was going to get a present for Jack. She went to get her piggybank"

both these demons would be introduced, and each would provide a partial (and complementary) explanation for the line "She needed money" if it subsequently appeared; yet no assertion that she needed money would be made, and consequently DSP's understanding of this text would be unsatisfactory. Charniak proposed that in circumstances such as these, demons should be permitted to interact with other demons: such an approach

---

*1 The notation "?PERSON" indicates the use of PERSON as a uPlanner variable: pattern matching results in (temporary) assignment to this variable.

is preferable to adding further demons which, for example, look for various ways of getting money. But Charniak noted that for this some classification of demons would be necessary, since in the case of

> "Janet was going to get a present for Jack. She was going to the store."

both the 'GET PRESENT' and 'GO TO STORE' concepts would suggest a reason for needing money, yet both reasons taken together seem insufficient evidence for asserting that Janet does in fact need money. Only in a case where the text gives both a reason for and a consequence of needing money, do we have sufficient evidence. In DSP, the necessary classification of demons was never provided, though it was implicitly present in the actions the demon would perform when running. As will be seen in section 7.2.1, Rieger did classify the inferences made by his system; similarly, AD-HAC employs a classification of inference types which is discussed in section 8.5. Both classification schemes are considerably more complex than the simple cause/result division suggested by Charniak.

7.1.3) DSP's approach to reference determination

The conversion from input format to assertion format involves the creation and use of uPlanner tokens to represent the referents of noun phrases, and the creation of additional assertions concerning each token. Thus, the NP "a cake" in the sentence "Janet baked a cake" is represented by the token CAKE1 in the assertion derived from the input,

(BAKE JANET1 CAKE1)

and additional assertions are created which concern that token, eg.

(IS CAKE1 CAKE)

As illustrated in the preceding section, the NP "Janet" is represented by the token JANET1, and several further assertions involving JANET1 are generated.

In the case where previous sentences have been processed, NPs will often be used which refer back to previously mentioned objects or people, and it is naturally desirable to have a single token which represents this referent. However, NPs in later sentences will not necessarily refer back in this manner, and so the problem arises of determining when two NPs do corefer and when they do not.

DSP's first step along the path to determining referents is the construction for each NP of a possible referent list (PRL). The PRL contains all those tokens generated so far which could possibly corefer, including others from the same sentence, subject to linguistic constraints on pronoun coreference proposed by Lees and Klima.

Finding possible referents is a complex process. DSP takes note of the use of definite and indefinite reference, and uses special heuristics in the case of over-specified noun phrases. Essentially, this boils down to using all the descriptive information available to prune the PRL until continuing would eliminate at once all the remaining items in the PRL. Pruning is therefore stopped, and the remaining item(s) are deemed to be those sought. The PRL is then augmented by a newly created token in case the new NP does not refer back to a previous token.

In the case of pronouns, the PRL contains all the previously generated tokens which might be referred to by that pronoun, excepting those referents forbidden by linguistic considerations. A uPlanner variable is then created, and the PRL is associated with that variable by using uPlanner's "restriction" facility; this means that the variable is restricted in the range of tokens it will match; in this case, only tokens mentioned in the variable's PRL can subsequently be matched with the variable.

NPs other than pronouns are also given a PRL. In cases where the PRL is empty, a new token is created. (In section 7.1.2, I was assuming this to be the case, and so used JANET1 and CAKE1.) Where there are possible referents, a restricted variable is used in the same manner as for pronouns.

When demons are applied to the assertions, there may be a match between these restricted variables and some acceptable token. This relies on the specification of demons, where a demon's own variables may be assigned values when the demon is created: such a demon is effectively tailored to recognise assertions specific to individual people or objects. If this occurs, the restricted variable is assigned the value it was matched against, and this assignment becomes permanent. By this means, the pronominal reference - or definite reference, or whatever - is determined. The situation may arise where two restricted variables are matched against one another, in which case both are further restricted. In general, demons are assumed to have sufficient knowledge to determine pronominal reference. When no demon can uniquely determine a referent for such a token, DSP will pick the most recent referent. This corresponds to the observation that a pronoun is more likely to refer to the most recently mentioned possible referent than to any other.

7.1.4) Summary

DSP attacked the problem of how real world knowledge could be applied to the task of understanding stories. The theoretical model of how this should be done corresponded in many respects to a partial implementation in Micro-Planner; but since the implementation was never completed, it is difficult to judge in concrete terms just how successful the model was.

One of the fundamental tenets of DSP was that inference should be performed "on the fly", rather than waiting until some specific task needing inference - for instance, resolving an anaphora or answering a question - had to be performed; and this assumption underlies most of the subsequent work on inference.

One of the principal features of DSP was the way inferential activity contributed to reference determination, though it should be noted that there is a serious asymmetry in the way DSP permits demons to determine reference, since cases may arise where several demons are looking for similar patterns: in these circumstances, the first such demon which is matched against the assertion makes its own idiosyncratic decision, regardless of the preferences of other demons which are later in the queue. Charniak defends this asymmetry on the grounds that demons are applied in inverse chronological order of addition, saying that the most recently added demon, and hence the first to be applied, has the most up-to-date information and so can make the most informed decision. This argument tacitly assumes that all demons are of equal importance, which is not the case. As will be seen in section 7.2.3, Rieger's scheme overcomes this problem by associating confidence ratings with inferences; and a similar method is used in AD-HAC.

DSP's major weakness was its rigid distinction between "real facts" and "potential facts". uPlanner assertions were used to represent the state of the world, and demon patterns loosely corresponded to possible inferences. This impelled Charniak to propose that demons should be classified, and this is an idea which was taken up in Rieger's work (as described in the next section), and which has been pursued also in AD-HAC. Apart from motivating this idea, however, the distinct status of demon patterns meant that DSP's inferences had to be rather shallow, and consequently DSP could only understand very simple stories.

7.2) Rieger: MEMORY

Rieger attempted to construct a model of human memory and human inferential processes, using Conceptual Dependency as the representation language. The resulting program, MEMORY, was linked with Riesbeck's sentence analyser and Goldman's sentence generator, to form MARGIE. The resulting system was therefore a more complete model of human understanding than was Charniak's, and was certainly easier to popularise.

CD was introduced to meet the need for a language-free formalism for representing meanings in a canonical fashion. One of the advantages of CD is that the primitive ACTs do indeed capture significant (conceptual) generalisations, and so do promote the ideal of canonical representation. However, the very low-level nature of the representation did emphasise several problems which Charniak has been able to sidestep. For instance, the CD representation of

"Janet baked a cake"

is considerably more complex than Charniak's "(BAKE JANET1 CAKE1)", even allowing for the extra constituents, like tense markers, mentioned earlier. In CD, the representation is centred upon the use of conceptual primitives, specifically primitive acts and primitive states, and the inference mechanism is presented with a set of conceptualisations, each of which is a very simple proposition based on an act or state. Part of the problem of drawing inferences from such data is the problem of noticing larger patterns covering several members of the set, and the failure to overcome this eventually proved to be MEMORY's downfall; nevertheless, CD is a better motivated representational language than the ad-hoc collection of terms used by Charniak, as is shown by the continuing Yale work on story understanding based on it.

7.2.1) MEMORY's basic machinery

Where DSP uses uPlanner assertions, MEMORY uses superatoms to represent conceptual information. These superatoms are Lisp atoms, generated as needed, whose property lists hold conceptualisations in an internal format which is equivalent to, but more easily worked with than, the list format used for input purposes. Superatoms also have a confidence rating and an interest rating, and various types of cross-reference information.

MEMORY's inference mechanism comprises a set of inference molecules, each associated with an individual primitive. These inference molecules specify the inferences to be made from a conceptualisation which employs the corresponding primitive. (A significant part of the justification of CD's collection of primitive terms, given in [Schank, 1975], is that the appearance of a certain primitive in a conceptual representation signals the appropriateness of a set of inferences.)

The inference molecules are applied to conceptualisations, which may be either new input to MEMORY or previous inferences. When an inference molecule specifies a new inference, it has to specify also a confidence rating, an interest rating, and a class of inference. The inference itself is specified as a conceptual pattern, and is converted to a superatom: the inference class is used, together with the superatom on which the molecule is working, to indicate where the inference came from - its reasons. Rieger used a total of 16 inference classes, as shown overleaf.

A newly created superatom, corresponding to an inference drawn from some old superatom, was also given confidence and interest ratings which depended upon both the ratings given by the inference molecule, and the weights given to the old superatom. If the inference was sufficiently interesting and sufficiently certain, it would in turn be used for the generation of new inferences. When no inferences (superatoms) met this criterion, MEMORY saw if there remained any unresolved references, pronouns for example, for input text. If such references did remain, MEMORY tried to resolve these references, or at least to eliminate some possibilities. When a reference was resolved, some of the existing inferences often became identical, and so they were merged. In the process of merging, they usually became more certain and more interesting, and sometimes became eligible for serving as the basis of further inferences. When this happened, the MEMORY program interleaved inference with reference: Rieger called this the inference/reference relaxation cycle.

7.2.2) MEMORY's machinery in action

The input to the MEMORY program was in the form of conceptualisations, also called conceptual diagrams, which represented the meanings of individual sentences. Initially these were produced by hand, but when MEMORY was incorporated into MARGIE they were produced by Riesbeck's sentence analyser. The conceptualisations were then converted into superatoms, while the PPs (Picture Producers, corresponding to concrete nouns) were stored in "PP-memory" for use in reference determination, which is further described in the next section. The appropriate inference molecules were applied to the newly created superatoms, causing further superatoms to be created and equipped with confidence ratings, interest ratings, and reasons.

When a superatom was created, its name would be added to a list of appearances for each of the PPs in the corresponding conceptual diagram, termed the PP's occurrence set. This provided an indexing scheme between PPs and relevant conceptual information, which was exploited, using a simple intersection search, to discover when an inference was being repeated: the program could therefore simply ensure that each conceptual diagram was uniquely stored.

'Repeated' inferences - ie. the same inference drawn from different processes - were of great importance to MEMORY. Just as DSP needed to permit demons to interact with other demons in order to assert information which provides a link between one part of the text and another, so MEMORY attempted to find inferences which were predicted consequences of one sentence, and precursors of another. Rieger's elaborate classification of inferences permitted such links to be recognised reliably. Further, where DSP's demons, triggered by individual lines of the story, looked for specific assertions, MEMORY applied inference processes to its own inferences in a less directed manner, and so could construct much longer

<u>MEMORY's classification of inference types</u>

i) specification inferences: What are the missing conceptual components in an incomplete graph likely to be?

ii) causative inferences: What were the likely causes of an action or state?

iii) resultative inferences: What are the likely results (effects on the world) of an action or state?

iv) motivational inferences: Why did (or would) an actor want to perform an action? What were his intentions?

v) enablement inferences: What states of the world must be (must have been) true in order for some action to occur?

vi) function inferences: Why do people desire to possess objects?

vii) enablement-prediction inferences: If a person wants a particular state of the world to exist, is it because of some predictable action that state would enable?

viii) missing enablement inferences: If a person cannot perform some action he desires, can it be explained by some missing prerequisite state of the world?

ix) intervention inferences: If an action in the world is causing (or will cause) undesired results, what might an actor do to prevent or curtail the action?

x) action-prediction inferences: Knowing a person's needs and desires, what actions is he likely to perform to attain those desires?

xi) knowledge-propagation inferences: Knowing that a person knows certain things, what other things can he also be predicted to know?

xii) normative inferences: Relative to a knowledge of what is normal in the world, determine how strongly a piece of information should be believed in the absence of specific knowledge.

xiii) state-duration inferences: Approximately how long can some state or protracted action be predicted to last?

xiv) feature inferences: Knowing some features of an entity, and the situations in which that entity occurs, what additional things can be predicted about that entity?

xv) situation inferences: What other information surrounding some familiar situation can be imagined (inferred)?

xvi) utterance-intent inferences: What can be inferred from the <u>way</u> in which something was said? Why did the speaker say it?

chains of inferences than could be obtained with DSP. However, since many of MEMORY's inferences were not logically entailed by its input texts, inferences drawn from them were on shaky ground. In DSP, the patterns of demons represented potential facts, while assertions represented "the truth": when the same demon pattern, or potential fact, was suggested independently from two lines of a text, DSP was happy to assert that it was true. In MEMORY, the superatoms played both the role of assertion and the role of demon: the association of confidence ratings with superatoms permitted the system to handle uncertain information.

As just noted, MEMORY used its own inferences to draw further inferences. Consequently, processing the successive sentences of a story led to the creation of a huge number of inferences, forming, in Rieger's words, a cloud of inferences surrounding each input conceptualisation (*2). The connectivity of a text was seen to lie in the collision of such clouds: when the same inference was drawn from two successive sentences, some link had been found: that inference, and the chains of inferences leading to it from both directions, enjoyed increased plausibility. Since the most convincing such chains employed predominantly causal links, where A "leads to" B "leads to" C and so on, Rieger called them causal chains. The understanding of a text was then seen as the construction of such causal chains.

As described in the next chapter, section 8.5.2, AD-HAC is also able to construct causal chains: but, as pointed out there, it does not place the same emphasis on these as did MEMORY.

7.2.3) MEMORY's approach to reference determination

Noun phrases tend to be represented in CD as "Picture Producers". MEMORY accordingly placed the corresponding conceptual representations in "PP-memory", keeping also a record of all the superatoms in which a given PP played a role. This was the PP's occurrence set mentioned above, and had to be recorded explicitly; in DSP, by contrast, uPlanner automatically recorded the set of assertions which mentioned a given token, so there was no visible analogue of Rieger's occurrence sets.

PPs describe what is known about a real-world object by using a descriptive set, a transformation into CD notation of any adjectives, possessives or relative clauses used in the text to qualify a head noun, together with information about the selected sense of the noun itself. Rieger, following Charniak, constructed a list of possible referents for any new PPs, whether pronominal or not, when they occured in the input conceptualisations; and, like Charniak, he applied a ranking to the various possible forms of modification, ensuring that over-specified noun phrases were permitted to refer back to previously mentioned objects.

To determine the reference for a given PP, MEMORY reordered the descriptive set of that PP, and then conducted an intersection search using each descriptor in turn, until either

(a) Only one candidate remained
In this case, the single remaining candidate was assumed to be the correct referent. Any unused descriptors in the descriptive set were

---

*2 The number of inferences produced was so large that MEMORY was usually artificially restricted to producing inferences only of selected types when running. This restriction could be lifted if more memory were available.

taken to be new information, and were added to the chosen candidate's descriptive set.

(b) The descriptive set had been exhausted
When the set of descriptors had failed to select one candidate over the others, two heuristics were used to break the tie: if one candidate had been explicitly referred to more recently than the others, that was chosen; if this heuristic failed, but one candidate had been "touched" by the inference processes more recently than the others, that was chosen; if still no single referent could be selected, then a new token was created having as its descriptive set the intersection of the descriptive sets of all remaining candidates. Such a token could then be used in the same way as proper references. These created tokens were placed on a list called !REFDECISION, which was scanned when the inference-drawing process came to a halt.

(c) Using the next descriptor would eliminate all remaining candidates
If there were no candidates at all - that is, the first element of the descriptive set would eliminate all existing tokens - then a new concept was created with the new descriptive set, and also placed on a list called !REFNOTFOUND; if part of the descriptive set had been processed, however, the same operations as for case (b) above were performed. If one candidate was selected by the 'recency' and 'touched' heuristics, that candidate was deemed to be the referent, and its descriptive set was augmented with the unused items from the new descriptive set.

However, where DSP permitted a demon to resolve a pronoun reference, MEMORY was much more cautious. The inference process continued apace, manipulating pronominal PPs in fundamentally the same manner as it manipulated PPs with determined references, for as long as there were superatoms which met the criteria of certainty and interest which determined whether they should be used for further inference production. When no more inferences could be drawn, because no superatoms fulfilled these criteria, the reference processes came into play. Generally, they succeeded in determining some reference, and the inference processes then continued. This was the inference/reference relaxation cycle.

7.2.4) Summary

MEMORY was able to understand more complex stories than DSP ever could, since it handled inputs with less explicit connectivity. It was able to figure out much more for itself; however, it remained unable to handle realistic stories. In particular, stories about stereotyped activities - such as trips to a restaurant - were found to require the construction of inordinately long causal chains to explain how one event might lead to another. Because there is no absolute criterion for determining whether a given causal chain is the correct one without comparing it with others, the search for a causal chain must be breadth-first: and this is a recipe for combinatorial explosion.

Realistic stories are often concerned with stereotyped situations; to understand such a story requires the deployment of knowledge which is specific to that situation. The sort of knowledge MEMORY had was of a quite different nature, being largely concerned with the preconditions of events, and the results of events occurring. In short, MEMORY's knowledge of the world was at too low a level to handle these stories.

The model of inference embodied in MEMORY was ultimately abandoned in favour of "scripts" and "plans", which were oriented towards the higher-level knowledge which MEMORY lacked. However, the primacy of scripts and plans in subsequent programs, such as SAM and PAM, restricts the applicability of these programs to stories of a particular type, as discussed in section 7.3. Furthermore, many problems, such as script selection and script termination, have appeared.

AD-HAC returns to the style of inference used in MEMORY, and seeks to overcome its problems in a different manner. The similarities between AD-HAC and MEMORY are many: inference processes are keyed on the primitive ACTs and STATEs; individual inferences are assigned certainty and interest ratings; inferences may (though only occasionally) be used to generate new inferences; and the inferences made by the system are classified, though using an altered classification. The problem of handling "scripty" stories is solved, however, by associating inference procedures with particular concepts, augmenting the ordinary low-level inferences with higher-level domain-specific inferences.

7.3) <u>The genesis of scripts and plans</u>

Though MEMORY's performance was superior to that of DSP, this raised false hopes about the complexity of the stories which could be handled. As noted above, many realistic stories are about highly stereotyped situations, and require the reader to apply extensive background knowledge. Roughly, the necessary knowledge is <u>episodic</u> rather than <u>semantic</u>. If MEMORY were presented with such a story, the requisite causal chains would be very long indeed. In seeking these chains, enormous numbers of inferences would have to be created, and most of these would in fact be irrelevant.

<u>Scripts</u> were proposed as a solution to this problem. In essence, scripts were seen initially as precoded inference chains: when a MEMORY-like inference mechanism finds itself dealing with a stereotyped situation, it can exploit the information encoded in the script to predict further inputs, and thereby obviate the need to seek explanatory causal chains by the more conventional bottom-up approach. This of course assumes that the inference mechanism can recognise that it is dealing with a stereotyped situation, and this asssumption has been the root cause of many problems in script-handling systems: indeed, script identification remains a serious problem to this day, and has prompted a great deal of development in the theoretical notion of scripts. Thus, scripts are no longer merely precoded inference chains, but are equipped with "precondition headers" and "default tracks", and are classified as "situational scripts", "personal scripts" and "instrumental scripts". The rationale for these developments is given in [Schank and Abelson, 1977].

Other stories which MEMORY could not handle, for very much the same reasons, were concerned with more general situations than scripts could handle: scripts are relatively specialised knowledge structures, dealing as they do with trips to restaurants, taking girls to cinemas, starting cars and so on. In particular, stories concerning goal-oriented behaviour are as much beyond the abilities of a script-based system as they were beyond MEMORY's: it is impractical to provide a script which lists the various ways in which a person might seek to raise a large sum of money, for instance. Understanding such goal-based stories involves first recognising the goals of a protagonist in a story, and then recognising his <u>plans</u>, that is, the actions he undertakes in pursuit of those goals. Therefore, a plan-recognition mechanism has to be provided.

A large part of the work of a script-based understander involves matching story statements with predictions generated by a script. When a script is deemed to be applicable to a story, the script is instantiated: as the next section will make clear, this involves determining which track is being followed, and which actors and objects in the story correspond to which scriptal roles. The use of plans is more complex: a plan will often suggest a number of scripts which an actor might employ in order to achieve his goals; the instantiation of plans is correspondingly more elaborate. Plans operate at a higher level than do scripts: there is even speculation that script-like knowledge in humans is constructed when the same plan steps are chosen repeatedly in real life.

In order to explore how scripts and plans could be applied to story understanding, the SAM and PAM programs described below were developed. As a way of testing their interpretations, each used Lehnert's QUALM [Lehnert, 1978] in order to be able to answer questions about the texts they dealt with. The input conceptualisations for each program were provided by Riesbeck's English Language Interpreter, ELI (*3), and sentences were generated in a variety of languages - English, Spanish, Chinese, etc. - by various upgraded versions of BABEL, Goldman's original sentence generator. However, there has been no attempt to construct a program capable of handling both scripts and plans: SAM and PAM are only able to handle script- and plan-based stories respectively. (PAM does use SAM as a subroutine, but still expects its stories to be plan-based: the whole is less than the sum of its parts.)

7.3.1) SAM (Script Applier Mechanism)

SAM was built to explore the role of scripts in the understanding of stories, and to explicate the nature of scripts themselves [Cullingford, 1978], [Schank et al, 1975]. Its input was a set of conceptual diagrams, constructed by ELI, and its output was a giant causal chain which could be inspected by other programs to answer questions, or to summarise the text, or to translate the text into other languages.

Each script was given a name, eg. '$RESTAURANT', and contained a precondition header, one or more maincon, and one or more track. Thus, for the $RESTAURANT script, the precondition header would state that the visitor to the restaurant was hungry; eating some food in the restaurant would be a maincon; and this maincon, together with inferences about seating, ordering, paying, tipping and leaving, would constitute a track.

A script would often have more than one track, and in such cases one of the tracks, corresponding to the most normal set of inferences, would be designated the default track: for the $RESTAURANT script, the default track would be that just outlined. Other tracks in a script would provide sets of inferences appropriate to relatively infrequent occurrences, and diverge from the default track at well-defined points. For instance, where the default track specifies paying-tipping-leaving, another track would specify dishwashing-leaving.

Each script had an associated set of "script roles", which served as variables in an analogous fashion to the variables in DSP's demons. When a

---

*3 ELI is a later version of Riesbeck's analyser described in Chapter 4.

script was instantiated, some of the variables were bound to particular PPs: thus, when $RESTAURANT was instantiated during the processing of the text

"John went to Leone's"

one of the script roles, say '&PATRON', would be bound to the PP representing "John". Subsequently, conceptualisations which matched onto one of those specified in the script, but which used the pronoun "he", could exploit the binding of the variable &PATRON to determine the referent of "he".

Similarly, some script roles would be associated with default values when the script was instantiated, permitting subsequent definite references to be understood, even when the referent had not already been mentioned. The role of scripts in rendering such references acceptable is well demonstrated by a contrasting pair of examples given in [Schank and Abelson, 1977]:

a) John went to a restaurant.
He asked the waitress for coq au vin.
He paid the check and left.

b) John went to a park.
He asked the midget for a mouse.
He picked up the box and left.

In example (a), our knowledge of restaurants provides a context in which the definite references "the waitress" and "the check" make sense. Our knowledge of parks, however, does not provide any comparable context in example (b) for "the midget" or "the box" (even though it is perfectly reasonable for midgets and boxes to be in parks); consequently, the text seems odd: the definite reference leaves us somewhat confused.

Scripts are, as mentioned earlier, relatively specialised knowledge structures. They are of no utility at all outside their domain; therefore a script-based system needs a variety of scripts if it is to handle a variety of stories. This immediately introduces the problem of script selection.

One of the problems tackled by SAM was that of determining which of its scripts to apply to a particular text, initially assuming that only one script will be appropriate. It was assisted in this by MEMTOK, a module which served much the same function in SAM as PP-memory did in Rieger's MEMORY. MEMTOK internalised the representations of PPs, and further suggested that particular scripts might be appropriate. Thus, for the sentence

"John went to a restaurant"

the script $RESTAURANT would be suggested, simply on the grounds that the word "restaurant" appeared in the sentence. SAM noted such advice, but demanded that some match within the body of the script should be found before it was prepared to instantiate the script: thus, though MEMTOK would make the same suggestion for

"Fuel oil was delivered to the restaurant"

there was no corresponding conceptualisation stored in the script, and so MEMTOK's simplistic advice was ignored.

The total number of scripts available to SAM was never very large, and so it was feasible, if necessary, to search for a matching conceptualisation in each script in turn. A later script-based program, FRUMP [DeJong, 1979], had a considerably larger number of scripts, but each was considerably less detailed than one of SAM's: furthermore, FRUMP was intended only to "skim" stories in order to produce quick-and-dirty summaries; if a particular sentence in one of FRUMP's inputs could not be understood, it was simply ignored.

Even referring only to situations covered by the small number of scripts SAM had, it was quite possible to construct stories which made fleeting reference to several scripts. The complexity of interaction needed between scripts seems unlimited; coherent stories can easily be constructed to relate arbitrary information from various scripts: SAM however had unsophisticated script switching operations.

One of the trickier problems for SAM was determining when a story deviated from a script, and in particular, determining when a particular script was of no further use in understanding a story. SAM applied several heuristics to overcome these problems, principally requiring fairly conclusive evidence that a script was applicable to a story before instantiating it: nevertheless, these problems were never adequately solved. A script-based story which took an unusual turn would frequently leave SAM nonplussed.

SAM's processing of a story consisted essentially of matching the conceptual representations of successive sentences to a prestored template, the script. In doing so, much of the generality of Rieger's MEMORY was lost, resulting in an inability to understand non-script-based stories. Further, SAM is so biased towards handling stories of the script-based variety that it is hard to imagine how the existing framework could be retained, but augmented to restore the lost generality.

Scripts nevertheless have definite merits. In particular, the ability to name scripts, and to refer to a large sequence of conceptualisations by naming the script, specifying the variable bindings, and indicating the track taken, is a valuable step towards a theory of forgetting. As Schank points out, forgetting is not a wholly bad thing: only by forgetting detail is it possible to remember the gist of a text; by such means as those sketched above, it would be possible to do so economically, and hence to infer some of the details later if required. Similarly, having names for such large-scale events - names, that is, which mean something to an understanding program - means that the fine grain of a conceptual representation can be bypassed in certain circumstances: the very representation language used by the system can incorporate these higher-level concepts. Provided that a symbol such as $RESTAURANT can ultimately be reduced to the primitive concepts of CD, such a facility is of great value. As will be seen in the next section, the ability to refer to a script as an entity can be very helpful in representing higher-level knowledge.

7.3.2) PAM (Plan Applier Mechanism)

PAM was built to determine how plans could help in the comprehension of goal-based stories, and to clarify the nature of plans [Wilensky, 1977]. PAM's input and output were in the same form as described above for SAM.

In order to understand stories about goal-directed behaviour, PAM used knowledge about themes to explain how goals might arise; the discovery of goals led to the suggestion of sketchy plans which might be employed in the furtherance of these goals. Each plan specified the goal it was expected to achieve, various conditions which had to be satisfied in order to proceed with the plan, and a set of planboxes which specified the options for each step of the plan.

Some texts explicitly describe a character as having a certain goal, but more frequently such goals have to be inferred. So, in the text:

"John loved Mary but she didn't want to marry him.
One day, a dragon stole Mary from the castle."

PAM used the 'LOVE' theme, which is explicitly stated, to infer that John wanted to marry Mary. The story, in this case, explicitly indicated that a related goal was absent, viz. that Mary did not have the goal of marrying John. When the second line of the text was processed, PAM inferred that Mary was in danger, and the LOVE theme predicted that John would want to rescue Mary.

For the example text above, John's goal of marrying Mary caused PAM to suggest a standard plan, getting the lady to church; this plan had a precondition, namely that the lady wants to marry the man, which in this case does not hold. PAM knew only the one plan for getting married, and it discovered it wouldn't work: so John's plan was blocked, and his goal was unachievable.

The second goal, John wanting to rescue Mary, caused PAM to suggest a second plan, quite independent of the first. This plan had the goal of achieving Mary's safety, and the planboxes involved

a) Getting near to Mary
b) Freeing Mary from the dragon
c) Taking Mary away again

These planboxes were themselves reminiscent of goals, but they also specified known means of achieving these goals, often by naming scripts. PAM was able to use SAM as a subroutine when it was trying to match inputs against scripts named as means of achieving certain goals.

If the story continued:

"John got on his horse...."

this could be recognised as part of a horse-riding script, which would be one of the specific means suggested by the first planbox. Similarly,

"....and killed the dragon."

could be recognised as a way of achieving (b) above.

Goal-oriented stories can be much more complex than this, especially when there are several actors in a story with conflicting goals. The interactions between the goals of one or more actors have been partially classified in [Schank and Abelson, 1977], and the resulting taxonomy involves:

i) Goal origin

Goals may have a thematic origin, as "John wants to marry Mary"; They may arise regularly and spontaneously, as in wanting to eat; they may arise as subgoals of other goals, as in John wanting to get near Mary; and they may arise in response to a crisis, as in John wanting to rescue Mary.

ii) Goal specification and substitution

The general goal of wanting to eat may be converted to a more specific goal of wanting to eat oysters; the goal of going to the pub may be substituted for the goal of going to the cinema.

iii) Goal suspension

Some goals are considered more urgent than others. When an urgent goal arises, a less urgent one may be temporarily set aside, but will typically be resumed when the more urgent goal has been either achieved or abandoned.

iv) Goal embellishment

When a goal has already been achieved, further goals may arise. Having acquired a good steak in a restaurant, for instance, the goal of acquiring mustard may arise.

In order to recognise the plans being used by a story actor, PAM needed to know about his goals, about how they might be achieved, and about how the plans used might affect other goals. In understanding a story, PAM therefore constructed goal fate graphs, showing how each goal originated, whether a particular goal succeeded or failed, and how goals related to one another according to the above taxonomy. It should be noted that all this related to the goals of one individual: though PAM had an impressive ability to tell back a story from several different points of view, it had great difficulty in appreciating the interactions between the plans of different individuals.

7.3.3) Review of scripts and plans

MEMORY's troubles arose because it lacked high-level knowledge. The knowledge it had was oriented towards the conceptual primitives of CD: the knowledge it lacked was of situations and behaviours. Scripts and plans were part of an effort to formalise these missing kinds of knowledge. They are specialised "frames", serving to organise knowledge in chunks which may be used both for recognition and for prediction. Scripts may be seen as frames prescribing sequences of actions, while plans often suggest sequences of scripts, but may descend to the level of specific actions if necessary.

There can be no doubt that higher-level knowledge, such as that which scripts and plans are intended to capture, is a necessary ingredient of a general story understanding system. The experimental systems SAM and PAM were constructed partly in order to elucidate the nature of scripts and of plans; it is unfortunate that the processing which they apply is so specific to their domains, and that no program has been built that can operate in both domains.

One of the problems with these systems is that their understanding of a text is wholly dependent upon these higher-level knowledge structures: the low-level inferencing performed by Rieger's MEMORY has been almost completely abandoned (though Lehnert's QUALM does do some of this when answering questions about the stories read by SAM and PAM).

7.4) Conceptual overlays

EX-SPECTRE-1 [Rieger, 1975] was intended to interpret sentences in contexts set up by previous sentences; particularly, to reach different interpretations for the same sentence in different contexts, and to explain why some sentences are judged "peculiar" in certain contexts.

The fundamental feature of EX-SPECTRE-1 was its use of conceptual overlays, whose main components were a cloud of expectancies and a set of expectancy selectors. A line in a text was intended to activate such a conceptual overlay, which would then perform various tests to set the "saliency" of the various expectancies.

The conceptual overlays were implemented as ternary discrimination networks: from each test, there would be three branches, for "Yes", "No" and "Don't know". (In most cases, the "Don't know" branch would take the same route as one of the other tests, and so provided a default path.) When a path was taken, the saliencies of nominated expectancies were set, and then the test (if any) at the end of the path was performed.

The set of expectancies generated by previously activated overlays constituted the context in which a subsequent sentence was to be understood, while the saliencies determined how strong an expectation was, rated roughly 0.1 to 0.9. A matching operation was intended to determine whether an expectation was fulfilled or not, but some of the expectancies served a different function: these were switchers, and a match with one of these would bring in a new overlay.

For example, if "X steals Y from Z", this activates the THEFT overlay, where tests like

Is Y valuable?
Does X know who Z is?
Are Z and X friends?
Is it difficult for Z to replace Y?
Had Z insured Y?

would be used to determine the probability of such future actions as

X will sell Y to somebody else.
X will avoid Z.
X will deny stealing Y.
Z will replace Y.
Z will report the theft to the authorities.

There would also be "switchers" corresponding to "X is happy" and "Z is upset"; these, if matched in subsequent sentences, would cause the introduction of further overlays, "PRACTICAL JOKE" and "ATONEMENT" respectively. (These switchers might be thought of as demons.) These switchers explicitly state what next sentences would be peculiar.

AD-HAC, like EX-SPECTRE-1, uses networks to control its inference processes, and also uses "certainty ratings" which are analogous to "saliency" and to MEMORY's "confidence". The details of the networks, and the use made of the certainty ratings, are however different; in particular, AD-HAC's approach to the "Don't know" answer to tests is to take both "Yes" and "No" paths, and to mark the resulting inferences as incompatible with each other. This is discussed further in section 8.2.2.2.

7.5) Summary of previous work on inference.

The preceding sections have illustrated the earlier work on inference which has been most influential in the design of AD-HAC's inference mechanism, which is described in the next chapter. Charniak's DSP set a trend in the understanding of texts, by using "informal" inferences to provide the connectivity between individual sentences. Rieger's MEMORY expanded upon this, by constructing much longer chains of inferences with well-defined causal links. However, realistic stories could still not be handled by these means, because their authors expect their readers to possess, and to be able to apply, knowledge about routine situations and about human goal-oriented behaviour, which is not explicitly mentioned in the stories. The notions of scripts and plans arose as part of an effort to characterise such knowledge, but their use requires specialised processing. The existing programs for applying this knowledge are not integrated: consequently, away from their specific domains, they understand less than MEMORY did.

While there has been much other work on inference, I have focussed upon those projects which used CD-like representations, because the inference mechanism used in AD-HAC has been inspired mostly by that work. Naturally, there are certain similarities with work in other paradigms. Wilks, for example, used chains of inferences to resolve difficult anaphoric references. Aspects of his translation system have been described in chapters 3 and 4. As mentioned there, the system was based upon "preference semantics", and one way of viewing preference, as applied by Wilks, is that the preferred meaning of a text is that interpretation which introduces the least new information. During the "extended mode" of processing [Wilks, 1975a], common sense inference rules are applied to the various semantic structures created by the earlier phases. The existing structures undergo a process of extraction, during which more tractable structures are produced; the inference rules are then applied, and are permitted to form chains of inferences. The shortest such chain, which corresponds to making the fewest assumptions, is then selected as determining the correct anaphoric referent. Wilks's system is radically different from Rieger's and Charniak's, however, in that it only uses these inference rules in order to solve specific problems - namely, to resolve "tricky" anaphoric references. However, the extension of this approach in other circumstances is discussed in [Wilks, 1978].

BELIEVER [Schmidt et al., 1978] is proposed as a psychological model of how beliefs, intentions and goals are attributed to an actor in response to simple descriptions of his actions. (BELIEVER is restricted to the plans of one actor.) It infers his goal, generates a plan for achieving that goal, and tries to match the descriptions of his actions onto the plan it has generated; it is also capable of revising the currently hypothesised plan if it does not fit the described actions.

BELIEVER maintains a "world model" which records facts about the world, and a "person model" which records the beliefs and desires of the actor. Plan generation begins when some goal for the actor appears in the person model: the plan grows by expansion of subgoals (and hence subplans) when some precondition is not satisifed in the person model. When complete, the plan becomes an expectancy structure, which makes predictions about the actions which will be described next. When a prediction is satisfied by an actual description, it becomes part of a "grounded plan", and subsequent plan steps are then expected.

When predictions are confounded, however, BELIEVER tries to revise its hypothesised plan. There are several rules which have been developed for this revision process, the rules being local to certain linkages of propositions: for instance, "Outcome" or "Subgoal". Revised plans are submitted to a set of "critics", which may constrain the order in which plan steps could be carried out, or may mark certain steps in the new plan as "will not be observed", or may reject the new plan altogether.

The basis of the psychological modelling is this Hypothesise/Revise cycle: Schmidt et al. argue that plan recognition is an ill-formed task, and therefore not amenable to the generate-and-test paradigm of artificial intelligence research.

Back within the CD paradigm, TALE-SPIN [Meehan, 1976] used plans to generate stories rather than to understand them. Plans are only one of several knowledge sources which have to be integrated by the program, though they do play the principal role because they provide motivation for events which the program can then relate. Meehan's work on story generation showed how diverse was the knowledge needed to perform well in this area: one is left with the suspicion that a program which merely understands stories, even when supplied with the additional knowledge provided in scripts, is able to conceal a great deal of ignorance. Ignorance becomes plainly obvious in story telling.

The next chapter describes AD-HAC's inferencer, which exploits many of the ideas whose development and implementation have been traced in this chapter. AD-HAC uses "inference networks" to specify inferences to be drawn from an input CDform. The networks contain both tests and actions: the tests are largely concerned with selecting a path through the network, while the actions are mostly inference specifications.

AD-HAC's inferences bear certainty and interest ratings, and they are also classified; the precise classification used has interesting consequences when inferences must be drawn from negated propositions, since the various classes of inference are modified in idiosyncratic ways.

One of the most important features of the inferencer to be described is the uniform treatment afforded by the use of inference networks to low-level (primitive-oriented) knowledge and high-level (object-oriented, script-like) knowledge: both are encoded and used in the same simple fashion.

Chapter 8: Inference 2: The AD-HAC inferencer

This chapter describes AD-HAC's inferencer, which associates inference networks with both conceptual primitives and classes of object. The inference networks apply tests to conceptual structures and to preexisting context, and generate appropriate inferences. Extensive cross-reference between individual inferences is maintained behind the scenes, and is exploited to compact the set of inferences which constitute the memory: this compaction leads naturally to referent identification, both for pronouns and for the answers to WH-questions.

Section 8.1 outlines the tasks which the inference mechanism is intended to perform, why inference is necessary, and what overall structure an inferencer should have to perform those tasks.

Section 8.2 explains how "inference nets" may selectively draw inferences from conceptual structures; how TEST and ACTION nodes are interleaved in these networks; how the appropriate networks are accessed; how they are traversed; and how they may influence the selection of pronoun referents.

The manipulation of conceptual patterns is discussed in section 8.3. The specification and matching of CDforms are among the most fundamental activities of the inference mechanism: for an inference to be drawn, a corresponding CDform must be specified; and for that inference to be useful to the process of reference determination, it must be internalised, that is, converted into a form which facilitates the comparison of CDforms, and the association of various forms of cross-reference information with individual CDforms.

Section 8.4 describes the use made by AD-HAC of "certainty ratings" and "interest ratings", which are numeric scores attached to the inferences when they are made, and which may be modified when similar inferences are merged.

The inferences themselves are classified as described in section 8.5. The classification which has been developed, besides being applicable to the construction of causal chains, also provides a simple mapping between the inferences drawn from affirmative sentences and those drawn from negated sentences, or sentences using the modals CAN, CANNOT.

Section 8.6 shows how the varied mechanisms contribute to the system's pronoun-resolution and question-answering abilities. These two tasks are perceived as being essentially the same: in either case, the determination of referents for WH-questions or for pronouns is accomplished by compacting the entire set of inferences drawn from the sentences constituting a story; and in either case, the inference nets may assist the pronoun resolution process by a process analogous to the specification of selectional restrictions on the conceptual level.

8.1) The task and some of its implications

The principal aim of this project was to explore the question of how understanding of a text might be achieved by a computer program, where "understanding" is judged by the ability to perform certain tasks. Foremost among these tasks is reference determination, an obvious and essential facet of human language comprehension which underpins our ability to answer questions and to resolve pronouns in discourse. AD-HAC was not intended as "a question-answering system", merely as a program which exhibited these abilities.

The production of inferences from some given conceptual pattern is not necessarily a proper task in its own right: only insofar as it enables, or facilitates, the answering of questions and the resolution of pronouns, is it a desirable component of a NL 'understanding' system. The assumption underlying this research has been that the mechanisms which permit reference determination do in fact depend critically upon inference. From this viewpoint, the processing required to resolve pronouns is very similar to that required for answering questions: an understander determines a pronoun referent by asking himself a question.

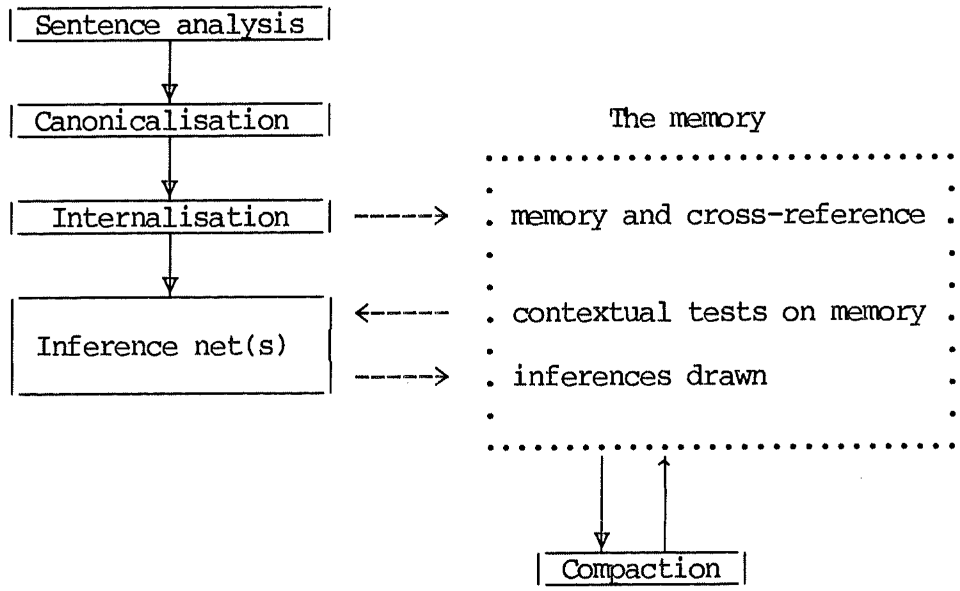
The inferences produced must be incorporated into some memory if they are to be useful: mere production of inferences solves no problems at all; rather, similarities between inferences drawn from different sources must be recognised. AD-HAC's representation of a text does not take the form of a giant conceptualisation; the memory is instead a network-like structure, with nodes corresponding to individual simple CDforms, and with varied types of link between these CDform nodes. Each CDform, whether produced by the sentence analyser or by the action of inference networks, bears extensive indexing and cross-reference information: it proves expedient also to give each such CDform a name for reference from other parts of the memory. The insertion of a new CDform into this memory is called internalisation, and is discussed at length in section 8.3.

Overall structure

The sentence analyser described in Chapter 5 feeds the inferencer with conceptual representations of the successive sentences of a text. The inferencer splits these representations into elementary propositions, "canonicalising" them as described in section 3.3.2; for each such elementary proposition, the main conceptual primitive - the ACT or the STATENAME - identifies the appropriate inference network. This network is then activated. These networks contain both tests and actions: the tests serve to determine a path through the network, inspecting both the structure of the CDform at hand (the driving conceptualisation or driver) and the global context (the state of the memory); the actions produce inferences, which are themselves incorporated into the memory.

When all the CDforms from the analyser have served for the production of inferences, the memory is compacted: the cross-reference information associated with the propositions stored in the memory indicates (among other things) which other propositions are similar, and what identifications would have to be made for two propositions to become identical. Compacting the memory involves collating this information, and determining which identifications should be made: these decisions are guided by certainty ratings associated with individual propositions; the inferencer attempts to merge propositions which agree with one another, and to avoid merging those which disagree. Since tokens representing pronouns and WH-elements will be among the suggested identifications, pronoun resolution and question answering are natural side-effects of the compaction process.

Below is a block diagram sketching this overall structure.



8.2) The inference process: inference nets

The inferencer described here uses a set of networks to produce inferences from a conceptualisation: each network is for a time totally in control of the inference process, and may make arbitrary inferences relating to the current driving conceptualisation. Many of the inferences made in fact correspond to presuppositions; in general, the inferences are idiosyncratic and non-deductive. These networks are initially keyed on the main primitive; thus there is a network devoted to ATRANS events, another for PTRANS events, and so on. The networks themselves may be viewed as a development of discrimination nets (*1). They have TEST nodes, which determine a path through the net, and ACTION nodes, which perform a variety of functions in addition to the main one of drawing inferences.

The selection of a path through a network is a simple way of making certain inferences dependent upon the local context. Meehan, describing his TALE-SPIN program, relates how the inference "GRAVITY DROWNED" was produced from "HENRY ANT FELL INTO THE RIVER": the principal cause of this was that 'x falling' was represented as 'GRAVITY PTRANSing x'; and an inference produced from a PTRANS event was that the ACTOR of the event changed location, as well as the OBJECT. Meehan's solution to this problem was to change the representation of "FALL" to use the CD primitive PROPEL. This seems completely wrong, since gravity always PROPELs things. A better solution is to make such an inference depend on the context (here, strictly local context), and this is achieved by the use of path-selecting tests. Other uses of tests are discussed in section 8.2.2.

This network formulation of the inference process proves extremely flexible: the basic networks - those keyed on conceptual primitives - perform a lot of bottom-up processing; others, often keyed on functions of objects, provide top-down processing corresponding to scripts and plans.

---

*1 Each network is represented as an association list, where the transition from one node to another is achieved by reference to the tag. These tags are numeric, simply to facilitate the writing of an independent FORTRAN program to produce plots showing the structure of the networks.

All of these networks have the same gross structure, differing in their roles but not in their form: therefore they can be processed in the same way.

8.2.1) Inference net ACTION nodes

The ACTION nodes of a network may do several things besides producing inferences. All basic networks begin by placing selected roles of the driving conceptualisation into registers, for easy access later on. Other operations include (i) declaring that two tokens must be distinct, and
(ii) defining a token in terms of some conceptual pattern

The uses of ACTION nodes will be illustrated using examples from the ATRANS inference network, whose first node reads as follows:

```
( 1 (ACTIONS
        (EXTRACT actor (EVENT ACTOR))
        (EXTRACT obj (EVENT OBJECT))
        (EXTRACT from (EVENT FROM))
        (EXTRACT to (EVENT TO))
        (EXTRACT-TIME-INFO)
        (DISTINGUISH from to)
        (DISTINGUISH from obj)
        (DISTINGUISH to obj)
        (DISTINGUISH actor obj)
        (INFER implicit 0.95 0.1
               ((STATE (STATENAME EXIST)
                       (THING (r to))
                       (TIME  (c timearound)))))
        (INFER implicit 0.95 0.1
               ((STATE (STATENAME EXIST)
                       (THING (r from))
                       (TIME  (c timearound)))))
        (INFER implicit 0.95 0.1
               ((STATE (STATENAME EXIST)
                       (THING (r obj))
                       (TIME  (c timearound)))))
        (INFER precondition 0.8 0.4
               ((STATE (STATENAME LOC)
                       (THING (r from))
                       (VAL (r to))
                       (TIME (c startspan)))))
        (GOTO 24)))
```

The function EXTRACT is used to place subparts of the ATRANS driving conceptualisation into registers. Thus, given the driver

```
((EVENT (ACTOR FRED)             ; "FRED TOOK A BOOK FROM HIM"
        (ACT ATRANS)
        (OBJECT BOOK1)
        (FROM DUMMY-MALE1)
        (TO FRED)
        (TIME (NAMED TIMEPOINT1))))
```

the first call to EXTRACT would place '(FRED)' in the register 'actor'. Similarly, 'obj' would hold '(BOOK1)' and so on.

The function EXTRACT-TIME-INFO then fills out a number of registers, allowing easy subsequent reference to the time of the driver, and various related times, such as the start- and end-points of a timespan.

The function DISTINGUISH is then called to specify to the pattern matcher that the fillers of certain registers can never be identified. In the present example, the matcher might otherwise consider 'FRED' as compatible with 'DUMMY-MALE1'; the call (DISTINGUISH from to) tells the matcher that this is not so. The other calls do not, in this example, have any significant effect.

Next, some inferences are proposed. The function INFER takes

(a) A class of inference - see section 8.5
(b) A certainty rating - see section 8.4
(c) An interest rating - see section 8.4
(d) A conceptual pattern specification - see section 8.3.1

In this example, many of the inferences are in fact trivial. The first inference made is of the class implicit, quite certain, and uninteresting:

```
(INFER implicit 0.95 0.1
       ((STATE (STATENAME EXIST)
               (THING (r actor))
               (TIME (c timearound)))))
```

The conceptual pattern specification in this example uses the contents of register 'actor', and also information supplied by EXTRACT-TIME-INFO, to build, in this case, the conceptual pattern:

```
((STATE (STATENAME EXIST)                        ; "FRED EXISTS" (*2)
        (THING FRED)
        (TIME (NAMED TIMESPAN1)
              (COMPARISON (-DURING TIMEPOINT1)))))
```

The other inferences made by this first node of the ATRANS network for our example sentence will be:

"THE MALE, FROM WHOM FRED TOOK A BOOK, EXISTS"
"A BOOK EXISTS"
"FRED WAS NEAR THE MALE, FROM WHOM FRED WAS TAKING A BOOK"

Finally, '(GOTO 24)' directs the inference processor to node 24 of the ATRANS inference net, a TEST node.

Another use of ACTION nodes - (ii) above - is the definition of a token in terms of some conceptual pattern in which it appears. For instance, if Fred takes something from Bill (say), it is desirable to ask if Bill realises that Fred, or at any rate somebody, has taken something from him: in the latter case, the 'somebody' has no real existence as a separate entity, but is defined in terms of Bill's realisation. The same definition process is used to prevent the pronoun reference mechanism from trying to identify quantified expression tokens with story objects.

8.2.2) Inference net TEST nodes

The test nodes in inference nets fall into two basic classes:
(i) Simple predicates concerning the driver or its components

---

*2 Strictly, "FRED EXISTED AT THE TIME HE TOOK THE BOOK"; the generator, perhaps mercifully, does not give that much detail about times.

(ii) Complex predicates concerning the context in memory
The simple predicates are concerned with the elements of patterns, rather than patterns as wholes. Thus, they are concerned with the equality of role fillers, the features of fillers, the structure of embedded conceptualisations (*3), the presence of certain roles, the relative sizes of objects, the commercial value of objects, etc. The complex predicates involve searching the memory for whole patterns matching a given conceptual structure. These two types will be discussed separately.

8.2.2.1) Simple tests: selecting paths, assisting pronoun resolution

The simplest tests are concerned with the equality of fillers and the features of fillers. Simple examples of these, in the PTRANS net, are:

```
(12 (TESTS (EQR actor object)     ; if the registers actor and
           13                     ; object hold the same filler,
           15))                   ; advance to node 13, else to 15

(20 (TESTS (IS actor SOLID)       ; if filler of register actor
           22                     ; has feature SOLID, advance to
           QUIT))                 ; node 22, else leave the net.
```

In some situations, it is possible to declare that a sequence of these tests must not have a given outcome. Eg:

i) In an ATRANS event, ACTOR = FROM or ACTOR = TO
ii) In an MTRANS event, ACTOR must have feature HUMAN
iii) In a PROPEL event, OBJECT must have feature CONCRETE.

When a sequence of tests gives an impossible answer, the inference net issues a complaint, which can be used to constrain potential pronoun referents. This is not the main technique used for pronoun resolution, though it is the case that most complaints are provoked by pronoun occurrences: exceptionally, complaints may indicate that the sentence analyser has built a semantically inconsistent structure.

An example of a 'complaining' node might be:

```
(12 (TESTS (IS actor HUMAN)
           13
           (COMPLAIN 12)))
```

meaning, if register 'actor' holds a token with the feature HUMAN, advance to node 13; otherwise issue a complaint which is interpreted as suggesting retrying node 12. (In general, there may be an arbitrary number of TEST nodes to be retried.) When this complaint is issued, the register 'actor' will be inspected to see if it holds a pronominal token. If the offending token is a pronoun, the inference process for the current conceptualisation is suspended until all the conceptualisations in the representation have been either processed to completion, or suspended in a similar manner. The original inference process is restarted by finding potential referents for the pronoun which now give new results for one of the indicated TEST nodes, and reentering the network at the appropriate node, thus generating more inferences.

Finding these potential referents involves scanning the cross-reference information associated with any inferences which involve the pronominal

---

*3 Embedded conceptualisations may occur even after canonicalisation, when a CDform has roles such as MOBJECT which require them. Canonicalisation only separates out those embedded conceptualisations headed by FOCUS.

token, so it is desirable to draw some inferences before any complaint arises: this is partly the reason for the first ACTION node of the ATRANS net inferring that the 'FROM','TO' and 'OBJECT' elements all "exist". Also, it is useful to have some indexing between tokens and the inferences in which they appear.

Thus, these complaints assist the process of pronoun resolution by specifying the combinations of features of acceptable referents. This mechanism is only applicable to 'IT' or 'THEM' pronouns, since the pronouns 'HIM' and 'HER' will already have the feature HUMAN, and so will give the same results for any feature test as would, say, FRED. These are handled by the more general mechanisms described in section 8.6.2.

The other simple tests - presence of optional roles, relationships between objects, typical values of objects, etc - have no incidental uses for pronouns. Their function, like the primary function of EQR and IS, is to alter the path through the net, or to vary the strengths of inferences (see section 8.4). Additionally, the predicate EQR, which tests for equality of the contents of two registers, will "DISTINGUISH" the contents if they are not immediately equal. This too proves useful in constraining the resolution of pronouns.

8.2.2.2) Complex tests: inspecting a wider context

Test nodes may also interrogate the memory by specifying a conceptual pattern. This is done by the function MBELIEVE: the method of specifying conceptual patterns is the same as that used for specifying inferences.

In general, when a question is asked of the memory, no existing inference will match the pattern specified - although further inference may provide some answers. If it were assumed that the absence of a match indicated that the question should be answered negatively, the entire inference process would become unstable, since different English constructions can easily cause the initial conceptualisations to be processed in different orders. (Similar observations hold if the default answer is affirmative.) In any case, it is not always possible to determine whether a proper match has been found, since temporal relationships in particular may not always be decidable.

There are several possible strategies for situations of this sort. One option would be to "wait and see": this would involve leaving some demon to pick up the trail if an answer was ever found, and possibly leave another demon to assume some default answer if too much time elapsed. (This approach is taken in the Word Expert Parser, albeit in a different domain. The Word Expert Parser is discussed in section 10.3)

An alternative is to provide an immediate default: if the answer is not known at the time the question is asked, take one path rather than the other. This is Rieger's method in the "conceptual overlays" used by EX-SPECTRE-1; there seems to be no method for undoing the damage if the answer later turns up.

The approach taken in AD-HAC is to take <u>both</u> paths: the inference process branches, and all the potential inferences are made. If the test ever becomes decidable, then some of the inferences made may be invalidated, and are expunged; if the test remains undecided, then some sets of inferences are marked as <u>incompatible</u> with others. Some degree of reversibility occurs here too, because it may be that a contextual question is answered because the inferences resulting from that answer give good agreement with other, independently derived, inferences.

The incompatibility of sets of inferences drawn from different assumed answers to these tests is one of the many forms of cross-reference information maintained by the system: sections 8.3.3 and 8.5.4 return to the topic of incompatibilities.

8.3) The storage of conceptual patterns in memory

The conceptualisations produced by the sentence analyser, and indeed by the inferencer itself, have a syntactic structure as well as conceptual content, and these two facets are independent. The syntax is designed to simplify the interfaces between the various components of the system, whilst the conceptual content is designed to facilitate the processes of inference. Section 8.1 indicated that conceptual patterns needed to be named, indexed, and cross-referenced. Section 8.2.2.1 showed, in gross outline, one use of this index and cross-reference information. For these purposes, the inferencer transforms the syntax of the conceptualisations.

The storage of conceptual patterns, and the maintenance of various forms of cross-reference information, are essential to the approach taken by the inferencer. Section 8.3.1 describes how the inference networks <u>specify</u> conceptual patterns; section 8.3.2 then explains how these patterns are <u>matched</u> with other patterns which already exist in the memory; and section 8.3.3 shows how <u>cross-reference</u> information is gathered and stored. It might be mentioned here that the matching process itself is responsible for gathering one of the most critical forms of cross-reference information, namely <u>specifications</u>: these are used later to indicate which propositions might be merged during the compaction of the memory, as described in section 8.6.

The indexing of conceptualisations is based largely upon the main conceptual primitive, usually the ACT or STATENAME. When a new inference is made, its main primitive is used to select a list of other existing conceptual patterns based upon that primitive, and a match between the new pattern and any of the old ones is sought. Exceptionally, it may be that an identical inference has been made already, in which case no further action is needed. More often, it is necessary to <u>internalise</u> the new conceptual pattern. This is done by creating a Lisp atom, say ATRANS4 (*4) for some ATRANS-based event, and storing the conceptual content on the property list of this atom. It is convenient to call such an atom a <u>proposition</u>. The new proposition is added to the list of patterns based upon the main primitive, so that later patterns may match it. Also, for each token appearing in the pattern, the proposition is added to a list of propositions in which that token appears. Finally, the newly created proposition is cross-referenced by using information about partial matches.

8.3.1) Pattern specification

Conceptual patterns are specified by the function INFER, and also by the function MBELIEVE. A simple example was given in section 8.2.1. In more

---

*4 It was mentioned in Chapter 2 that this naming convention for nominal tokens was just a convenience in development and debugging,serving to make intermediate results intelligible; the same observations apply here.

complex cases, a pattern specification has two parts. The first part is a template, the second (optional) part is a filler list.

Various prefix characters may be used within the template to indicate that various substitutions should be made, and any part of the template which does not have such a prefix is copied directly to the new conceptual structure. The prefix characters are: r,c,v,$.

The prefix 'r' is used to indicate that the contents of a register are to be inserted into the conceptual structure. Thus, in the example given earlier, (THING (r actor)) uses the contents of the register 'actor', and is copied over as (THING FRED).

The prefix 'c' is used to create conceptual time representations. Its use must be preceded by a call to EXTRACT-TIME-INFO, which sets up various registers holding parts of the driver's time specification. These registers are:

```
time .......... the driver's complete time structure

startpoint .... if the time is a TIMESPAN, its TS
                if it is a TIMEPOINT, that TIMEPOINT

endpoint ...... if the time is a TIMESPAN, its TF
                if it is a TIMEPOINT, that TIMEPOINT
```

For example, suppose the driver is

```
((EVENT (ACTOR FRED)
        (ACT ATRANS)
        (OBJECT BANANAS1)
        (FROM FRED)
        (TO MONKEYS1)
        (TIME (NAMED TIMEPOINT1)
              (COMPARISON (BEFORE *NOW*)))))
```

Then calling EXTRACT-TIME-INFO will fill the registers as follows:

```
      time = ((NAMED TIMEPOINT1) (COMPARISON (BEFORE *NOW*)))
startpoint = TIMEPOINT1
  endpoint = TIMEPOINT1
```

Alternatively, given the driver

```
((EVENT (ACTOR FRED)
        (ACT PTRANS)
        (OBJECT FRED)
        (FROM DUMMY-PLACE1)
        (TO ZOO1)
        (TIME (NAMED TIMESPAN1)
              (TS TIMEPOINT1)
              (TF TIMEPOINT2))))
```

the registers would be filled thus:

```
      time = ((NAMED TIMESPAN1)(TS TIMEPOINT1)(TF TIMEPOINT2))
startpoint = TIMEPOINT1
  endpoint = TIMEPOINT2
```

When these registers are set up, the 'c' prefix can be used as follows:

```
(TIME (c beforespan)) yields (TIME (NAMED TIMESPANx)   (*5)
                                   (TF ~startpoint))   (*6)
(TIME (c afterspan))  yields (TIME (NAMED TIMESPANx)
                                   (TS ~endpoint))
(TIME (c timearound)) yields (TIME (NAMED TIMESPANx)
                                   (COMPARISON (-DURING ~time)))
(TIME (c startspan))  yields (TIME (NAMED TIMESPANx)
                                   (COMPARISON (-DURING ~startpoint)))
```

These commonly used time structures can thus be easily incorporated into the specifications of conceptual patterns, and may be referred to by the use of these mnemonics.

The prefix 'v' causes the substitution of the value of any Lisp expression: most commonly this is simply a free variable.

The '$' prefix is used in conjunction with the second element of a pattern specification, the filler list: it is used to prefix a number, which is interpreted as a subscript for the filler list. Thus, ($ 1) refers to the first element of the filler list, ($ 2) refers to the second, and so on. A concept (a Lisp atom) in the filler list causes a new instance of the concept to be created; if the first element is TIMEPOINT, for example, the instance TIMEPOINT3 might be created, and will be used in place of each occurrence of the subpattern "($ 1)" in the pattern specification. There are other ways of using the filler list, but I shall not describe them here: they are seldom used and their effects could be achieved in other ways.

The use of filler lists can be illustrated with an example from the MTRANS network. This call to INFER occurs when the driver corresponds to, eg, "FRED WROTE A BOOK". This motivation inference picks up "FRED", "BOOK" and "CONCEPTS" from the driver (via the registers), but needs to invent a new DUMMY-HUMAN and a new TIMEPOINT in order to express the idea that "FRED THOUGHT THAT SOMEBODY WOULD BENEFIT FROM READING WHAT HE WROTE"

```
(INFER motivation 0.8 0.5
      ((STATE (STATENAME MLOC)
              (INCP (r actor))
              (TIME (r time))
              (MOBJECT
                 (CAUSE
                    (ANTECEDENT
                       (EVENT (ACTOR ($ 1))
                              (ACT MTRANS)
                              (MOBJECT (r info))
                              (FROM (r to))
                              (TOCP ($ 1))
                              (TIME (NAMED ($ 2))
                                    (COMPARISON (AFTER (r time))))))
                    (RESULT
                       (STATE (STATENAME BENEFIT)
                              (THING ($ 1))
                              (VAL (HIGHERBY 2))
                              (TIME (NAMED ($ 2)))))))))
      ( DUMMY-HUMAN TIMEPOINT ))
```

---

*5 The notation TIMESPANx means a newly created TIMESPAN token

*6 The notation ~startpoint means the contents of the register startpoint

Taken together, the various methods of substituting partial structures into a pattern provide a convenient pattern specification mechanism. The most important feature of this scheme is that the pattern specifications are easily written and are easy to read later.

8.3.2) Pattern matching

As mentioned earlier, conceptual patterns have a main primitive, usually an ACT or STATENAME. With this primitive is stored a list of propositions based upon it. When a new pattern comes along, this primitive is identified and the relevant list of propositions is retrieved. Then the subsidiary roles of the new pattern are split into several groups, according to the type of matching appropriate for them. These groups are:

(a) Truth selectors, eg. (TRUTH TRUE),(ABILITY CANNOT)
(b) Simple nominals, eg. (ACTOR FRED),(OBJECT DUMMY-THING1)
(c) Procedurals, eg. (TIME (NAMED TIMEPOINT3)),(VAL (HIGHERBY 2))
(d) Embedded conceptualisations

The pattern matching procedures always check for compatibility, as explained below, as well as for identity between the elements of a pattern and the corresponding elements of previously stored patterns. When two patterns are not identical, but are in all respects compatible, the pairs of compatible elements are passed on as cross-reference information, as described in more detail in section 8.3.3 below.

Compatibility of truth selectors

In matching, the truth selectors are dealt with quite simply. The possible truth modifiers are
{TRUE,FALSE,CAN,CANNOT,TRUTH?,ABILITY?,UNTRUTH?,INABILITY?}
The matcher regards {TRUE,TRUTH?,CAN,ABILITY?} as compatible with one another, and {FALSE,UNTRUTH?,CANNOT,INABILITY?} likewise.

Compatibility of simple nominals

For simple nominals the rules governing the compatibility of two non-identical tokens are more complex. Firstly, they must not have been marked as distinct (by the functions DISTINGUISH or EQR); secondly, they must either be instances of the same nominal, or one of them must be a pronominal token having features compatible with the other.

These rules permit MONKEY1 to match MONKEY2, by virtue of their being instances of the same nominal - both having the property (ISA MONKEY) - and permit DUMMY-MALE1 to match FRED, since DUMMY-MALE1 is a pronominal token with the single feature MALE. These rules do not however permit NUTS1 to match PEANUTS1, because neither of the rules given above apply. This is a weakness of the currently implemented program: in order to overcome this problem, it is necessary to introduce some further classification of objects, organised along the lines of a thesaurus.

In the implemented program, components of patterns are classified as "simple nominal" by default: ie. if they do not fall into any of the other classes. Thus, in particular, the handling of superprimitives is performed by this part of the pattern matcher. Section 2.3.2 in the chapter on representation mentioned the introduction of superprimitives. The use of conceptual primitives carries significant benefits, allowing many fine distinctions of meaning to be represented. This can however be a disadvantage if all messages encoded in these primitives must be highly precise and explicit. For example, we may wish to infer that somebody who discovers that somebody else has stolen something will feel badly towards the thief, but there are several forms of bad feeling. Suppose "Fred discovers that Mary stole his wallet"; do we infer

"Fred dislikes Mary"
"Fred hates Mary"
"Fred is angry with Mary"
"Fred finds Mary repulsive"
"Fred loathes Mary"

The use of superprimitives permits the primitives corresponding to these "bad feelings" to be grouped together. In pattern matching, a superprimitive is regarded as compatible with any of the primitives which it groups together. The existing superprimitives, and the groups of related primitives, are shown below. (*7)

SOUND	*LAUGHTER*,*SPEECH*,*SONG*,*NOISE*
SENSOR	*EYE*,*EAR*,*NOSE*,*PALATE*,*SKIN*
NEGEMOTION	*HATRED*,*ANGER*,*DISLIKE*,*LOATHING*,*REVULSION*
POSEMOTION	*LOVE*,*TOLERANCE*,*LIKING*,*ADMIRATION*,*ATTRACTION*
AGENCY	*GRAVITY*,*PRESSURE*,*MOMENTUM*,*COHESION*,*ADHESION*, *EFIELD*,*MFIELD*
DIRECTION	*NORTH*,*SOUTH*,*EAST*,*WEST*,*UP*,*DOWN*
QUANTITY	*EVERY*,*MANY*,*SOME*,*FEW*,*NO*
LOT	*EVERY*,*MANY*
LITTLE	*FEW*,*NO*
PART	*TOP*,*BOTTOM*,*MIDDLE*,*END*,*INSIDE*,*OUTSIDE*, *HIGHPART*,*LOWPART*

This shows, incidentally, that a single primitive may match more than one superprimitive. For example, *EVERY* will match both **QUANTITY** and **LOT**. The contexts in which these primitives and superprimitives may appear are, of course, restricted. For instance, **SOUND** and its associated primitives may only appear as either the SOUND role of a SPEAK act, or as the TO role of an ATTEND act.

Compatibility of procedurals

The matching of procedurals is concerned with temporal information, and also with the scales used to represent many adjectives. Two non-identical times are seen as compatible, regardless of whether they are points or spans of time, if it is impossible to prove that one time is before the other: this can be an expensive operation, particularly since in almost all cases different time tokens are compatible. The internal representation of the time information has been tailored to make this operation as cheap as

*7 Items surrounded by single asterisks are primitives, those surrounded by double asterisks are the superprimitives possible, and the checking of temporal compatibility is the last, and hence least frequent, matching operation performed.

The scales used for the representation of most adjectives involve the use of a number to indicate a point on the scale. For instance, the SIZE scale takes a VAL in the range -10 to 10; various adjectives correspond to subranges, as follows:

MINUTE	-10 to -8
TINY	-7 to -5
SMALL	-4 to -1
BIG	1 to 5
HUGE	6 to 8
ENORMOUS	8 to 10

The pattern matcher does not insist that the numbers given match exactly, but instead allows some degree of slop by permitting a match between two numbers which differ by less than 3.

An alternative form of these scales is used to indicate a change of state. Thus, the verb "TO GROW" might be represented as a SIZE scale, with (VAL (HIGHERBY 2)), and "TO SHRINK" might be represented with (VAL (LOWERBY 2)). In pattern matching, these forms are permitted to match any number, and a HIGHERBY can match another HIGHERBY with some slop as above.

Compatibility of embedded conceptualisations

Embedded conceptualisations can occur in several situations. For instance, any MTRANS has a MOBJECT role which is an embedded conceptualisation, and any EVENT may have an INST role. These embedded conceptualisations are treated as patterns in their own right, going through pattern matching and then internalisation. These processes return a proposition token which has some cross-reference information attached to it; this in particular refers to the compatible propositions. When a pattern with an embedded conceptualisation is being matched against an existing proposition, the match will proceed if the existing proposition has a compatible embedded conceptualisation.

The matching proceeds in several stages:
(1) The main primitive is identified, and the list of propositions based on that primitive is retrieved.
(2) The conceptual roles are split into the groups mentioned above.
(3) Embedded conceptualisations are recursively matched and internalised.
(4) The truth selectors are used to prune the possible matches.
(5) The simple nominals are considered, one at a time. For example, the ACTOR might be considered first, then the OBJECT, etc.
(6) If the pattern is a scale, the VAL is considered.
(7) If embedded conceptualisations are present, the corresponding propositions are considered.
(8) Time tokens are checked for compatibility.

There is a final check to ensure that no token must be regarded as compatible with more than one other token. If this check were omitted, the structures

```
((EVENT (ACTOR DUMMY-MALE1)           ((EVENT (ACTOR FRED)
        (ACT ATRANS)                          (ACT ATRANS)
        (OBJECT BOOK1)                        (OBJECT BOOK1)
        (FROM DUMMY-MALE1)                    (FROM GEORGE)
        (TO FRED)                             (TO FRED)
        (TIME (NAMED TIMEPOINT1))))           (TIME (NAMED TIMEPOINT1))))
  "HE GAVE FRED THE BOOK"               "FRED TOOK THE BOOK FROM GEORGE"
```

would be regarded as compatible: this problem can appear in many forms, but the check described here traps all of its manifestations.

8.3.3) Cross-referencing

The cross-reference information attached to propositions is a critical element of the entire inference apparatus. Drawing inferences, and even expressing these inferences in English, is a limited, nay futile, exercise unless inferences made from different sources can be compared and perhaps merged. The comparison of conceptual structures is done by the pattern matcher, as described in the preceding section. The results of these comparisons are passed on to the rest of the system as cross-references between the propositions. Cross-referencing serves a variety of functions, as described below, all of which serve in particular to promote the system's major activities of pronoun resolution and question answering.

The cross-reference information is added when a conceptualisation is internalised: suppose we had the two conceptual patterns shown below, and the second one, ATRANS3, was being internalised:

```
 ATRANS1: "HE TOOK A BOOK FROM BILL"   ATRANS3: "FRED TOOK IT FROM BILL"

  ((EVENT (ACTOR DUMMY-MALE1)           ((EVENT (ACTOR FRED)
          (ACT ATRANS)                          (ACT ATRANS)
          (OBJECT BOOK1)                        (OBJECT DUMMY-THING7)
          (FROM BILL)                           (FROM BILL)
          (TO DUMMY-MALE1)                      (TO FRED)
          (TIME (NAMED TIMEPOINT1))))           (TIME (NAMED TIMEPOINT2))))
```

The pattern matching procedures described above would have detected that the following pairs of items were compatible (but not identical) (*8):

In ATRANS3 (new)	In ATRANS1 (old)	Role(s)
FRED	DUMMY-MALE1	ACTOR,TO
DUMMY-THING7	BOOK1	OBJECT
TIMEPOINT2	TIMEPOINT1	TIME

Therefore, the specifications attached to ATRANS3, the new proposition, indicate the similarity with ATRANS1, and the substitutions which would have to be made in ATRANS3 for these two propositions to be identical, thus:

```
(ATRANS1 ((FRED . DUMMY-MALE1)
          (DUMMY-THING7 . BOOK1)
          (TIMEPOINT2 . TIMEPOINT1)))
```

ATRANS1 would also be given the "reverse" specifications

```
(ATRANS3 ((DUMMY-MALE1 . FRED)
          (BOOK1 . DUMMY-THING7)
          (TIMEPOINT1 . TIMEPOINT2)))
```

---

*8 This discussion assumes that these pairs are in fact compatible.

These specifications are used for many purposes: proposing pronoun referents, evaluating pronoun referents, answering queries generated within the system, and determining when two propositions have become identical.

These specifications, recording the similarities between different propositions, are provided by the pattern matching procedure. The KRL pattern matcher, described in [Bobrow and Winograd, 1977], can be used in a similar fashion if the resources available are limited. Their matcher takes a pattern and a single datum, and they say:

> Resource limitation and pinpointing of further problems: In a case where the processing so far has not produced a definite answer, the matcher should be able to return specific details in addition to the result of "don't know yet". Given the problem of matching (which Owns (a Pet)) against Mickey, with sufficient resources ..... it could answer "Yes". With less resources, it could answer "Yes, if (a Dog) matches (a Pet)", and with still less, "Yes, if Pluto matches (a Pet)".

The specifications returned by the matcher in AD-HAC may be seen as a "Yes, if..." answer. The representation language used does not contain embedded descriptions in the way KRL does, which makes the application of deductive procedures inappropriate for AD-HAC's matcher. The most germane difference, however, is that in KRL the matcher attempts to establish references as soon as possible; in AD-HAC, the possibility of reference is noted at the time a match is attempted, and the entire set of possible references is later used to determine what identifications should be made.

There are other forms of cross-reference information attached to propositions and to tokens. When a proposition is created to hold a conceptual pattern, the name of that proposition is also added to the list of appearances of each token which occurs in that proposition. Suppose we had just the propositions ATRANS1 and ATRANS3 shown above, the appearances of various tokens would be as follows:

DUMMY-MALE1	(ATRANS1)
FRED	(ATRANS3)
BOOK1	(ATRANS1)
DUMMY-THING7	(ATRANS3)
BILL	(ATRANS3 ATRANS1)
TIMEPOINT1	(ATRANS1)
TIMEPOINT2	(ATRANS3)

This 'appearances' information is used, for example by the complaint-handling process introduced in section 8.2.2.1, to retrieve the set of propositions which may be relevant in making some decision about a token; when handling complaints, the specifications attached to these propositions are scanned to find potential referents for pronominal tokens.

When an inference is made, it is assigned an inference class, and this is also recorded as a form of cross-reference. As explained in detail in section 8.5 below, most inference classes are paired up with their complements, for use in mutual cross-reference. Thus, suppose the driver is ATRANS1, and a resultative inference is POSS1, then the information

    (resultative . ATRANS1)

is added to POSS1, and the complementary information,

    (causative . POSS1)

is added to ATRANS1. Cross-references of this kind are called inf-links.

Whenever a conceptual pattern is internalised, a proposition is created which names that pattern and stores the conceptual content on its property list. To facilitate the cross-reference process, an inverse proposition is also created, which differs by having the opposite truth value. Thus, when ATRANS3 is created as shown above, the inverse proposition ATRANS4 is also created having (TRUTH FALSE). The two propositions are linked together by having the property 'inverse'. Similarly, ATRANS1 would have an inverse, ATRANS2. While the provision of these inverses does facilitate the cross-reference process, it is rather clumsy and complicates various other parts of the program; eventually, it is hoped that these inverse propositions can be eliminated.

Assumptions and incompatibilities

As mentioned in section 8.2.2.2, contextual tests in inference networks result in both paths through the net being taken, and the resulting sets of inferences being marked as incompatible with one another. This is another form of cross-reference.

When a contextual test is made, using the function MBELIEVE, the conceptual pattern is internalised, and both branches of the network are followed. When following the "yes" branch following an MBELIEVE test, the proposition is added to a global list called 'assumptions'. Similarly, when the "no" branch is followed, the inverse proposition is added to this list. Whenever an inference is made (*9), the contents of the 'assumptions' list are copied to the assuming property of the resulting proposition, and this proposition is added to the assumption-indicates property of each of the assumptions.

Each inference produced in this manner is dependent both upon the current driver, and upon the current set of assumptions; in general, when a contextual test is made, disjoint sets of inferences are produced depending repectively on an affirmative and a negative answer to the test. If independent evidence is found which supports such an assumption-dependent inference, this support can be seen as preferring one answer to the test over the other. Hence these contextual tests may be answered indirectly, because of the inferences to which they lead. Also, if one such dependent inference finds support, the inferences dependent on the other answer to the test lose some credibility. To facilitate this interplay between the competing sets of inferences, it is desirable to have some direct reference between the elements of different sets, particularly since the paths through the inference network may later rejoin: in such a case, relatively complex computations would have to be repeatedly performed. Consequently, propositions depending on one answer to a test rather than another index all the propositions which assume the alternative answer. This form of cross-reference is called incompatibilities, and is also used when multiple inferences of particular types are drawn from a single driver. For example, several motivations may be inferred for an actor performing a particular act, yet usually only one of these motivations actually applies. This is discussed further in section 8.5.4 below.

The 'assumptions' mechanism is also used when an inference network is resumed after a complaint was issued by a simple IS test, as briefly described in section 8.2.2.1, and more fully described in section 8.6.1

---

*9 When an inference is expressed in English, the first assumption (if any) is also picked out, and the generated sentence has the form "IF ...,THEN ...".

below. In such a case, a proposition is created which asserts identity between a pronominal token and some candidate referent. This proposition is placed on the list of assumptions, and the inference process continues, marking each subsequent inference as assuming this reference for the pronoun, as above, and recording each on the assumption-indicates property of the proposition which asserts the identity of reference. Subsequently, as shown in section 8.6.1, these inferences are scanned, and the agreement between them and other independently-derived inferences is exploited to select the best referent for the pronoun.

8.4) Certainty and Interest ratings

The specification of inferences is accomplished by use of the function INFER, which, as indicated in section 8.2.1, requires a certainty rating and an interest rating to be specified, along with a conceptual pattern and a class of inference. These ratings are usually given as numbers in the range 0.1 to 0.9. When the conceptual pattern given to INFER is internalised, causing the creation of a proposition as described in section 8.3 above, this proposition is assigned a certainty weight and an interest weight on the basis of the numbers cited in the original call to INFER, chosen in an ad-hoc manner defended below.

A high certainty rating indicates that the proposition is believed to be true, whilst a high negative rating indicates that the proposition is believed to be false. Intermediate numbers indicate that the proposition is more or less strongly believed. The value 1 is taken to mean that the proposition is an unshakeable belief.

A high interest rating indicates that it may be fruitful to produce further inferences, using this proposition as the driver. As will be seen in section 8.5.2, these ratings may be systematically modified in order to encourage the construction of causal chains.

The propositions corresponding to canonicalised constituents of the conceptual structure delivered by the sentence analyser are initially weighted at 0.99 for both certainty and interest. When inferences are drawn, in the simplest (and commonest) case (*10), the weights given to the inferred propositions are multiplied by the corresponding weights of the driver: thus, when chaining occurs, there is a progressive weakening of both the certainty and interest of remote inferences; this means, in particular, that the inferential process can be halted if a threshold is passed.

Whenever a proposition is created, an inverse is also created as described in section 8.3.3 above. Such an inverse proposition has certainty and interest weights complementary to those of the real proposition, and hence negative in the range 0 to -1: thus, if the proposition POSS1 is created, saying "FRED HAS THE BANANA" for instance, POSS2 will say "FRED HAS NOT GOT THE BANANA"; and if POSS1 has a certainty of 0.7, then POSS2 will have a certainty of -0.7.

When two propositions are merged, one of the operations performed is the calculation of new certainty and interest weights. The new weights should exhibit the following properties:
(a) The new weight should remain in the range -1 to +1

---

*10 More complex cases are dicussed in sections 8.5.1 and 8.5.2

(b) If one of the old weights was +1, the new weight should be +1; conversely, if one of the old weights was -1, the new weight should be -1. (If one of the old weights was +1, and the other was -1, the two propositions should not be merged.)
(c) If one of the old weights was zero, the new weight should be taken directly from the other old weight.
(d) If both old weights were positive, the new weight should be positive and higher than either of the old weights (unless one of them is +1, when (b) above applies)
(e) If one of the old weights is positive and the other is negative, the result should take its sign from the largest in magnitude.

The calculation of these new weights is performed by the function AGREE, which is reproduced below.

```
(DE AGREE (X Y)
   (COND ((ZEROP X) Y)
         ((ZEROP Y) X)
         ((OR (EQN X 1.0) (EQN Y 1.0)) 1.0)
         ((OR (EQN X -1.0) (EQN Y -1.0)) -1.0)
         (T (QUOTIENT (PLUS X Y)
                      (PLUS 1 (ABS (TIMES X Y)))))))
```

The use of numeric scores such as these is frequently, and mistakenly, criticised on a variety of grounds.

i: People surely do not have precise measures of their beliefs.
   The AD-HAC system does not pretend to be a model of how people understand stories, but is simply a computer system for understanding them. The use of numbers in such a machine-oriented model is perfectly natural, and in any case these numbers are peripheral to its reasoning abilities, providing it with some ability to represent information it needs to guide its operation. Some measure of the degree to which a proposition is believed is necessary, and it must be possible to systematically change this measure as more evidence for or against it is accumulated. Using numbers for this measure makes this modification quite simple.

ii: The selection of ratings ought to be guided by uniform criteria.
   The ratings are simply chosen to reflect how certain and important an inference seems to be, in the context established by having reached a particular point in the net. This is based on the implementor's common sense.

iii: The numbers are obviously important, but are they right?
   The precise values of these numbers does not seem to matter, at least in the texts on which the system has been tested. It has <u>never</u> been necessary to fine-tune the weights of any inference.

These ratings are used by the inference mechanism in several ways, which are outlined in the following sections.

8.4.1) Certainty ratings and proposition merging

When the conceptual representation of a text, or even of a single complex sentence, is used for the generation of inferences, it sometimes happens that precisely the same conceptual pattern is created for two or more inferences. Far more frequently, patterns are created which have an overall similarity, but which differ in some respect, particularly in relation to pronouns. For example, in processing the conceptual representation of the

sentence
"JILL TOOK BILL'S MONEY FROM HIM"
canonicalisation will produce two structures, say

```
((EVENT (ACTOR JILL)              ((STATE (STATENAME POSS)
        (ACT ATRANS)                      (THING MONEY1)
        (OBJECT MONEY1)                   (VAL BILL)
        (FROM DUMMY-MALE1)                (TIME (NAMED TIMEPOINT1))))
        (TO JILL)
        (TIME (NAMED TIMEPOINT1))))
```

both of which will first be internalised and then used for the production of inferences. Suppose the corresponding propositions are ATRANS1 and POSS1. One of the inferences drawn from ATRANS1 will have the form

```
((STATE (STATENAME POSS)
        (THING MONEY1)
        (VAL DUMMY-MALE1)
        (TIME (NAMED TIMESPAN1)
              (TF TIMEPOINT1))))
```

ie., the male from whom Jill took the money had the money previously. This inference will be internalised, and may end up as the proposition POSS3. The pattern matching procedures described in section 8.3.2 will have noted that this pattern is similar to the pattern of POSS1, so the proposition will have been given the <u>specifications</u>

```
(POSS1 ((DUMMY-MALE1 . BILL)
        (TIMESPAN1 . TIMEPOINT1)))
```

and POSS1 will also have been given appropriate specifications. Furthermore, the token DUMMY-MALE1 will have been given <u>appearances</u>

```
(POSS4 POSS3 ATRANS2 ATRANS1)      (*11)
```

(There will naturally be many more propositions, and much more cross-reference information, than I show here.) The specifications attached to POSS3 indicate that POSS1 may be saying the same thing as POSS3, if DUMMY-MALE1 can be identified with BILL, and if TIMESPAN1 can be identified with TIMEPOINT1. Since both POSS3 and POSS1 have a positive certainty rating, the system concludes that it would be desirable to merge them. In a realistic case, when many more inferences are made, with correspondingly more specifications, the system has to choose which similar propositions to merge: it does this by trying to merge those propositions which show the highest agreement. Thus, if two propositions are similar, and are strongly believed, this will act as an impetus to merge them. Conversely, if one proposition is strongly believed and another is disbelieved, the system will try to avoid merging them.

The certainty ratings, then, are used to determine which propositions the system should merge. By merging propositions, the system is able to resolve pronouns, answer wh-questions, determine many definite references, and establish many temporal sequences. More details are given in section 8.6.

8.4.2) <u>Interest ratings for controlling inference</u>

The system has the ability to use derived inferences themselves as drivers, to produce further inferences. This will generate chains of inferences with links of different kinds as indicated by the various networks. Similar chains were created by Rieger's MEMORY, as "expanding

---

*11 The appearances include the inverse propositions.

spheres of inference", and by Wilks [Wilks 1975a] when applying "common sense inference rules" (CSIRs) to resolve anaphora. Unfortunately, when this chaining occurs, there is a danger of combinatorial explosion: in Wilks's case, this was avoided by using inferences only in order to solve specific problems, ie. to resolve certain difficult anaphoric references; in Rieger's case, combinatorial explosion represented a serious problem.

The approach taken in AD-HAC involves assigning an interest rating to individual inferences, and drawing further inferences from only those inferences whose interest rating exceeds a certain threshold. This threshold is set somewhat arbitrarily at 0.3. Since the interest ratings of new inferences are always smaller than the weight given to the driver, this ensures that the inference process will halt. Further, inferences are in fact only chained when there is some causal relationship which is explicitly asserted in a text, and the precise details of this causal relationship need to be discovered.

Interest ratings can however be 'revived' in some circumstances. When propositions are merged, the interest ratings of propositions are modified in the same way as the certainty ratings, using the same AGREE function illustrated above. It is therefore possible for the merging operation to cause some inferred proposition to exceed the threshold, and therefore to be considered for the production of further inferences, even though both of the original propositions were considered insufficiently interesting.

8.4.3) <u>Interest ratings for assisting generation of sentences</u>

When the inferencer wishes to express some proposition in English, it expands the conceptual information stored on the proposition's property list into the external CD form, and passes this on to the sentence generator. The CD form may of course contain references to pronominal tokens - DUMMY-MALE1 for example - and the generator may find that it is impossible to cast the sentence in such a way that reference to this token is omitted. When this happens, the generator seeks further information which it can use to describe this dummy token, and any further information it receives will typically appear as a relative clause qualifying the dummy.

Upon receiving such a request from the generator for further information, the inferencer needs to pick some proposition which mentions the dummy token, and go through the same unpacking process to create a further representation in external CD form. Typically, there will be several such propositions, and the inference mechanism needs to pick one which will convey helpful information to a reader of the generated text, and preferably the most helpful one. The original proposition, though it clearly does mention the dummy, is not very helpful in giving further information. For instance, the relative clause does not serve any useful function in the sentence

"FRED GAVE MARY THE THING WHICH HE GAVE MARY".

(AD-HAC does not try to emulate bureaucrats!)

To pick the most helpful proposition, the system first looks at the <u>appearances</u> property of the dummy token, to find the propositions in which the dummy is mentioned. It then considers only those propositions which are believed, ie. have a positive certainty rating, and which have not already been passed to the generator for the production of this sentence. It then picks the proposition which maximises a function of the certainty and interest ratings. The interest rating plays the more important role in this

function, since very certain but uninteresting propositions lead to ludicrous results. For instance, the following sentence was produced in the early stages of this work:
THE MALE, WHO HAD SOME LEGS, LEFT THE PLACE FROM WHICH HE WENT TO THE ZOO.
Given a driver corresponding to "HE WALKED TO THE ZOO", a very certain inference is that "HE HAD SOME LEGS"; but this is a very uninteresting inference. Utilising interest measures to select a suitable qualifying proposition greatly enhances the quality of the English produced.

8.5) Classification of inference types

Irrespective of the precise mechanism used to draw inferences, it is clear that not all inferences are alike, and it is natural to try to formalise the ways in which inferences may differ: that is, to classify them. One obvious way in which inferences may be classified is on the dimension of time: some inferences will be about the past, others about the (relative) future. Within this broad classification, it is easy to distinguish finer subcategories: "preconditions" and "motivations", for example, will both concern the past, but are intuitively different types of inference.

The preceding chapter mentioned that Rieger's MEMORY program used a classification of inference types. Much of his classification has been carried over into the present work, but some of his classes have been dropped, and some new ones have been introduced. Additionally, as mentioned in connection with cross-reference in section 8.3.3 above, AD-HAC's inference types are paired up. In general, the pairings are rational, though in some cases type and anti-type are the same, and in some other cases pseudo-anti-types have had to be introduced. The types used, and their pairings, are:

precondition	:	enabledact	
enabledact	:	precondition	
resultative	:	causative	
causative	:	resultative	
motivation	:	plannedact	
plannedact	:	motivation	
motivator	:	motivated	
motivated	:	motivator	
instrumental	:	helping	
helping	:	instrumental	
precevent	:	follevent	
follevent	:	precevent	
normative	:	normative	(*12)
rewrite	:	rewrite	(*12)
actorfeature	:	never-use	(*13)
implicit	:	never-to-be-used	(*13)
exclusion	:	excludes	(*13)

---

*12 'rewrite' and 'normative' are their own anti-types
*13 'never-use','never-to-be-used' and 'excludes' are pseudo-anti-types

The names given to the various inference classes are hopefully indicative of their meanings. These meanings are clarified below, along with restrictions upon the types of proposition which may be linked by a given inference type, and with simple examples in each case.

(i) precondition type inferences are made when the driver is an EVENT, and relate to that event a prior STATE which must have been true for the event to have taken place. (*14)
For example, "JOHN GAVE MARY A BOOK" ---> "JOHN HAD THE BOOK"
Conversely,

(ii) enabledact type inferences are made when the driver is a STATE, and relate to it an EVENT which may take place when this state is true.
For example, "JOHN HAS A BOOK" ---> "JOHN MAY GIVE THE BOOK TO SOMEBODY"
Only the most obvious and potentially helpful enabledact inferences are actually made.

(iii) resultative type inferences are made when the driver is an EVENT, and relate a STATE which becomes true as a consequence of the event being performed to the driving event.
For example, "JOHN GAVE MARY A BOOK" ---> "MARY THEN HAD THE BOOK"
Conversely,

(iv) causative type inferences are made when the driver is a STATE, and relate to it an EVENT which may have brought about that state.
For example, "MARY HAD A BOOK" ---> "PERHAPS SOMEBODY GAVE THE BOOK TO MARY"

(v) motivation type inferences are made when the driver is an EVENT, and relate a goal-type STATE to it. Specifically, the actor of the event is inferred to have a goal of bringing about some state, and this state will normally correspond to one of the resultative inferences from the same event.
For example, "MARY TOOK THE BOOK" ---> "MARY WANTED TO HAVE THE BOOK"
Conversely,

(vi) plannedact type inferences are made when the driver is a goal-type STATE, and suggest EVENTs which may be performed to bring about the desired state. (*15)

(vii) motivator type inferences are made when the driver is an EVENT, and relate to it a STATE which might impel the actor to perform the event. These should not be confused with motivation inferences.
For example, "FRED ATE THE BANANA" ---> "FRED WAS HUNGRY"
Conversely,

(viii) motivated type inferences are made when the driver is a STATE, and relate to it an EVENT which the state might impel the actor to perform. (*15)

(ix) instrumental type inferences are made when the driver is an EVENT, and relate to it another EVENT which may be performed by the same actor, and at the same time, as a means of performing the driving event.
For example, "FRED TOLD MARY THAT ..." ---> "FRED SPOKE"
These instrumental inferences are clearly related to the INST role, which is an optional part of the representation of any event. Conversely,

(x) helping type inferences are made when the driver is an EVENT, and relate to it another EVENT which the same actor may be performing at the same time by means of the driving event.
For example, "FRED PUSHED THE TABLE" ---> "FRED MOVED THE TABLE"

---

*14 Precondition inferences usually have a very high certainty rating to reflect the strong expectation that they must have been true: but some are in fact rated lower, and this reflects a problem of representation.

*15 Currently, neither plannedact- nor motivated-type inferences are actually made by the implemented program. However, their use is anticipated.

Such inferences are seldom made: only the PROPEL network uses them.

(xi) precevent type inferences are made when the driver is an EVENT, and relate an EVENT which may have occurred earlier to the later event.
For example, "FRED WAS SICK" ---> "FRED ATE/DRANK TOO MUCH"
Conversely,

(xii) follevent type inferences are made when the driver is an EVENT, and relate to it an EVENT which may be expected to occur later.
For example, "FRED GAVE THE MONKEY A BANANA" --> "THE MONKEY ATE THE BANANA"
Both precevent and follevent inferences occur most frequently inside script-like inference networks, and permit many intervening STATE representations to be omitted.

(xiii) normative type inferences are made when it seems useful to make some inference which does not fall into any other category.
For example, "FRED'S CAR WAS STOLEN" ---> "FRED WILL INFORM THE POLICE"
This is a catch-all classification, and its existence indicates that further work on the classification of inferences is needed.

(xiv) rewrite type inferences are provided as a programming hack to overcome certain known weaknesses in the sentence analyser. They are used in two places only; the first corresponds to the use of phrases like "FRED'S SISTER", which is wrongly analysed using the state POSS; the second use handles sentences like "FRED ATE ELEPHANT", which is analysed as though "FRED ATE AN ELEPHANT"! Rewrite inferences specify a new conceptual structure to be handled in place of the current driving conceptualisation. This type clearly needs to be removed in system development.

(xv) actorfeature type inferences are made when the driver is an EVENT, and an idiosyncratic inference is made because of some special feature of the actor of the event.
For example, "GRAVITY PROPELs OBJECT" ---> "SOMETHING SUPPORTS OBJECT"

(xvi) implicit type inferences are always STATEs with the statename EXIST; they are always quite certain, very uninteresting, and made early in the inference nets.
For example, "FRED GAVE MARY A BOOK" ---> "FRED EXISTED"
The principal purpose of these inferences is to have some inference made about every object or person mentioned in a story, so that candidates can be found when a pronoun has to be resolved.

(xvii) exclusion type inferences are made when the driver is a STATE, and some other state can be deduced to be false.
For example, "FRED HAS A MONKEY" ---> "NOBODY ELSE HAS THAT MONKEY"
Such inferences can usefully restrict the pronoun reference processes.

8.5.1) Handling negation and ABILITY

The inference networks are written with simple affirmative drivers in mind: various inferences about preconditions of events, results of events, motivations, etc., are made in a straightforward way. However, many of these inferences would be invalid if such a simple driver is negated, or if the conceptual structure mentions ABILITY, and so the system must avoid making them. (*16)

The classification of inference types given above proves extremely useful in handling this problem, for some classes of inference seem to be independent of the truth of an assertion, whilst other inferences change in a systematic way. By exploiting these observations, the inference networks themselves need take no note of the truth value indicated for a particular

---

*16 The handling of negation is often discussed in terms of 'presuppositions'. [Wilson 1975] gives a detailed treatment of this approach.

driver: the function INFER itself changes the inferences made in a variety of ways, as follows.

When drawing inferences from a given driver, AD-HAC first inspects its TRUTH or ABILITY roles, and uses these to split the various inference classes temporarily into groups. These groups are called fullstrength, weakened, disbelieved and existentially-dependent. These names are mnemonic: "fullstrength" inferences have certainty and interest ratings computed in the standard way, by multiplying the ratings of the driving conceptualisation by the numbers given in the call to INFER; "weakened" inferences are given a certainty rating of only 50% of this full strength; "disbelieved" inferences have a negative certainty rating (their inverses are assigned 40% of the full strength computed as above). The final grouping, "existentially-dependent", has an effect on certain inferences involving "existentially-qualified" tokens: this is more complex, and is explained later.

Inference classes are temporarily assigned to these groups as follows:
(1) For a straightforward affirmative sentence, i.e. TRUTH=TRUE, all inference classes are fullstrength.
(2) If the driver has ABILITY=CAN (*17) then precondition, implicit and precevent inferences remain as fullstrength, while motivation and motivator inferences are disbelieved; all other inference types are ignored.
(3) If the driver has ABILITY=CANNOT, implicit inferences are weakened, precondition and precevent inferences are disbelieved, and all other types are ignored.
(4) If the driver has TRUTH=FALSE, then precevent and precondition inferences are weakened, motivation, motivator and enabledact inferences are disbelieved, and implicit inferences are existentially-dependent. The system may flag various tokens as 'existentially-qualified'; this flag indicates that the token, although a dummy, may never be permanently identified with another token, but must be permitted to match any other token at any time, provided the features are compatible. (This is one way in which the system deals with quantification.) Most commonly, this situation arises when the sentence analyser constructs dummy tokens to fill parts of a conceptual representation for a negated sentence, as described in section 5.3.4. For example, the sentence "THE BANANA WAS NOT EATEN" will be represented as

```
((EVENT (ACTOR DUMMY-BEAST1)
        (ACT INGEST)
        (OBJECT BANANA1)
        (TRUTH FALSE)
        (TIME ...)))
```

When this representation is constructed, the token DUMMY-BEAST1 will be flagged as existentially-qualified: with the flag, the conceptualisation may be loosely read as

"THERE IS NO BEAST WHICH ATE THE BANANA"

Without the flag, it would be

"THERE IS SOME BEAST WHICH DID NOT EAT THE BANANA"

When the inference network for INGEST is working on this representation, it will make an implicit inference that the ACTOR - ie. DUMMY-BEAST1 - exists. The effect of making implicit inferences 'existentially-dependent' is to make them sensitive to the presence of this flag. If the token whose existence is being predicated is marked with this flag, then the certainty

---

*17 Only EVENT conceptualisations may have an ABILITY role.

rating of the corresponding inference is reduced to 30 percent of its normal value; if the flag is absent, the full certainty rating is given.

Overall, this apparently simplistic handling of negation and ability proves highly effective, and the system's success in handling texts which involve these phenomena is encouraging. The most serious problem encountered so far is that there is no uniform way of handling normative inferences; but since this is a catch-all classification pending further research, it is hardly surprising, though regrettable, to find a lack of uniformity here.

8.5.2) Constructing causal chains

As outlined in the preceding chapter, Rieger's MEMORY program tried to link together the conceptual representations of the sentences in a text, and assumed that the identification of causal chains was a critical phase of this activity. Rieger also assigned inferences to various classes, and placed restrictions upon the types of conceptualisation which could be linked to form chains by inferences of given types. The theory underlying the MEMORY program essentially cultivated the nettle of combinatorial explosion, stating that the production of inferences continued indefinitely: when some sentence, a part of a text, caused the production of inferences, these too should be used for the production of further inferences, and so on ad infinitum.

The identification of causal chains was based largely on precondition and resultative inferences: EVENTs resulted in STATEs which enabled further EVENTs to occur. The MEMORY program therefore tried to find some sequence of intervening states and events which would explain how one event might follow another: such causal chains were seen as the fundamental explanation of the connectivity of texts, implying a particular view of the nature of text in general.

AD-HAC has a similar ability to construct causal chains, though they do not play the same central part in its processing of a story as they did in Rieger's work: the availability of long-range inference types, such as "follevent" and "precevent", and the formulation of script-like inference networks which make such long-range inferences, removes the need for almost all of this chain construction. However, chains may be constructed when the driving conceptualisation is based on the primitive CAUSE. First, the ANTECEDENT and RESULT parts of the driver are both subjected to inference. Many of the inference classes described in the preceding section may be seen as "forward-looking" or "backward-looking", that is, as predicting future and past events and states. To construct causal chains, AD-HAC specially emphasises the forward-looking inferences based on the ANTECEDENT part and the backward-looking inferences based on the RESULT, by increasing the interest ratings of inferences of these types. The direction categories are shown below: note that the normative, actorfeature and rewrite types fall into both camps, since these are not particularly well-defined inference types. Note also that implicit and exclusion inferences are not categorised, and are therefore not modified in this way.

forward-looking	backward-looking
resultative	causative
enabledact	precondition
plannedact	motivation
helping	instrumental
motivated	motivator
follevent	precevent
normative	normative
actorfeature	actorfeature
rewrite	rewrite

When a causal structure is being processed, the interest ratings of the ensuing inferences are increased in such a way that they in turn will be used for the production of inferences. This has the effect of increasing the probability that some connection will be found between the initial antecedent and the final result.

8.5.3) The role of inference types in updating states

It has been argued in [Schank 1975] and in [Rieger 1974] that the "meaning" of a primitive act lies in the set of inferences which are made about an event based on that act; and these inferences will usually be STATE inferences. On this view, the precondition and resultative inferences are primary: certain states are necessary for an event to occur, and certain other states result. Frequently, one of the resultative inferences changes a state referred to in one of the event's preconditions. Several examples may be given of this:

(a) a precondition of a PTRANS is that the OBJECT be located at FROM
    a result of a PTRANS is that the OBJECT is located at TO
(b) a precondition of an ATRANS is that the OBJECT be possessed by FROM
    a result of an ATRANS is that the OBJECT is possessed by TO
(c) a precondition of an INGEST is that the OBJECT be outside the ACTOR
    a result of an INGEST is that the OBJECT is inside the ACTOR

In each of these examples, the resultative inference can be seen as an update of the state referred to in the precondition inference: some state has been superseded by another state as a direct consequence of the occurrence of the event. AD-HAC's understanding of a text relies in part upon keeping track of states of the world, and recording the updating of one state by another. In all cases, the STATENAME role is constant. Thus, a POSS state will be updated by another POSS, and a LOC will be updated by another LOC, as in the examples given above. Similarly, the THING role is constant. So, the token found in the OBJECT role of an ATRANS is placed in the THING role of both the precondition and the resultative inferences, as shown below. Where the relevant state is a scale, and consequently the VAL is either numeric or of the form (HIGHERBY n) or (LOWERBY n), the THING must obviously be the same for both states for there to be any question of updating.

The situation diagrammed below illustrates the updating of a "precondition" state by a "resultative" state: the two states are both POSS states, and their THING is the same; the VAL has changed, and the TIME roles are consecutive. (Incidentally, this provides some justification for placing the thing possessed in the THING role, since the single generalisation then holds for all kinds of state.)

driver

```
((EVENT (ACTOR FRED)
        (ACT ATRANS)
        (OBJECT BOOK1)
        (FROM MARY)
        (TO FRED)
        (TIME (NAMED TIMEPOINT1))))
```

precondition

```
((STATE (STATENAME POSS)
        (THING BOOK1)
        (VAL MARY)
        (TIME (NAMED TIMESPAN1)
              (TF TIMEPOINT1))))
```

resultative

```
((STATE (STATENAME POSS)
        (THING BOOK1)
        (VAL FRED)
        (TIME (NAMED TIMESPAN2)
              (TS TIMEPOINT1))))
```

Several inference classes appear to be susceptible to the state updating phenomenon. A resultative inference may update any precondition, implicit or motivator inference made from the same event. When this occurs, the proposition for the resultative inference is given the property update-of, and the earlier proposition is given the property updated-by. For the ATRANS driver and POSS inferences given above, if the precondition inference is POSS1 and the resultative inference is POSS3, the update links will be:

(updated-by . POSS3) on POSS1
(update-of . POSS1) on POSS3

This is a further form of cross-reference between propositions, and, as will be shown in section 8.6.1, one which is particularly helpful in the resolution of some pronouns.

8.5.4) Incompatible inferences

An event may have many different preconditions, and it may have several results; yet usually only one of these results is the effect intended by the actor. Similarly, there are often several possible instrumental actions, but only one of them will actually be employed in the performance of some event. It is desirable for an inference mechanism to have access to information of this sort, since it may then build a more accurate and useful representation of a text.

The inference classes which display this phenomenon are causative, motivation, motivator, enabledact and instrumental. The system notices when more than one inference of any of these types is made for a single driver. When this happens, all the conflicting inferences are marked on each other's list of incompatibilities, which, as section 8.3.3 hinted, allows independent support for one of them to be regarded as evidence against all the others.

As a concrete example, with a sentence such as
"FRED WAS GIVEN HIS MONEY BY HIS MOTHER"
one of the causative inferences drawn from the POSS state, representing the information "HIS MONEY" as "HE HAD SOME MONEY", will suggest that he was given the money by somebody: another causative inference will suggest that he took the money from somebody; and these two inferences will be marked as incompatible. Consequently, when the first is matched up with the main proposition, "HIS MOTHER GAVE FRED SOME MONEY", the other causative inference is weakened.

8.6) Pronoun resolution, question answering, and definite reference.

The previous sections have described the basic machinery of AD-HAC; the way in which inferences are produced, how they are stored in memory, what information is kept with propositions besides their conceptual content, and the various forms of cross-reference maintained behind the scenes. This section shows how all that machinery contributes to the performance discernible from outside, viz. the resolution of pronouns and the answering of questions.

The problems of pronoun resolution and question answering are the principal focus of AD-HAC's inference mechanism. As stated in the introduction to this chapter, merely drawing inferences is not a particularly important activity, and even expressing these inferences in English is merely a shiny addition. However, if the system is able to answer questions and resolve pronouns, then it may be claimed that it has to some extent understood a text. Stronger proofs, such as summarising texts or applying knowledge gleaned from one text in the processing of another, are possible: but an ability to resolve pronouns and to answer questions is a sine qua non for an understanding system.

The machinery presented in the preceding sections gives the inference mechanism in AD-HAC the basic tools with which it may tackle these tasks. It turns out that pronoun resolution and question answering are essentially the same activity; similarly, the handling of definite references within a text, and the identification of temporal sequences, can be performed by the same basic mechanisms. The system effectively views pronouns as implicit wh-questions, and in either case the system's task is to select the referent from a set of candidate referents (or candidates for short). Consequently, I use the term pronoun to refer both to pronouns proper, and to WH-words.

The preceding sections have described the various forms of indexing and cross-referencing which are employed, and have introduced a number of special terms, which for convenience are collectively listed below.

(a) propositions are Lisp atoms which each store a conceptual pattern, and various cross-references, on their property lists. There is a straightforward mapping between the stored conceptual pattern and the corresponding external CD form.
(b) specifications are data structures stored with propositions, indicating similar propositions and the substitutions which would have to be made for the two propositions to become identical.
(c) appearances are lists of propositions held on the property list of tokens, showing, for each token, all those propositions in which the token is mentioned.
(d) certainty is a number stored with a proposition, showing how strongly that proposition is believed: the numbers range from +1 indicating absolute belief to -1 indicating absolute disbelief.
(e) inf-links, stored with a proposition, indicate the other propositions which have been inferred from it, and also the proposition(s) from which it was inferred. Each link gives both a connected proposition and the class of inference involved in making the connection.
(f) assumptions are lists of propositions, usually corresponding to uncertain answers from contextual tests, which are stored with each inference which depends upon such a test. Conversely, assumption-indicates is a list of these inferences which is stored with each provisional answer.
(g) incompatibilities are stored with a proposition, listing other

propositions which are in conflict with it. Conflict may arise in two ways: firstly, when several inferences of particular classes are made from the same driving conceptualisation - for example, several motivation inferences may be made, but they conflict with one another; secondly, when a contextual test is made, the inferences resulting from a provisional YES answer will be incompatible with those resulting from a NO answer.

8.6.1) Resolution of IT/THEM/WHAT pronouns by using complaints.

In some cases, the semantics of the conceptual ACT or STATENAME will demand that tokens appearing in a conceptualisation exhibit some feature, or one of some set of features. The simple test nodes in an inference net have the ability to inspect various characteristics of the driving conceptualisation, and to select appropriate paths through the remainder of the net. When a token does not carry appropriate features, one of the test nodes will issue a complaint. For example, the ATRANS network includes the test node numbered 24:

```
(24 (TESTS (IS from BEAST)
           25
           (COMPLAIN 24)))
```

This test checks the contents of the register from to see if its token - which has come from the FROM role of the ATRANS driver - has the feature BEAST. If so, the processor advances to node 25 of the net.

If the token does not have this feature, the other branch is taken, which issues a complaint. The form "(COMPLAIN 24)" indicates that the test at node 24 - the current node, as it happens - has given an impossible answer: that is, in order for the driver to be meaningful, the test at node 24 must succeed. How it is made to succeed is described shortly.

A more complex example of complaint generation occurs in the INGEST network, where the following cascade of tests may be found:

```
(3 (TESTS (IS obj BEAST)           4     8))

(8 (ACTIONS
      (INFER precondition 1.0 0.5
         ((STATE (STATENAME POSS)
                 (THING (r obj))
                 (VAL (r actor))
                 (TIME (c beforespan)))))
      (GOTO 9)))

(9 (TESTS (HAS-FUNCTION obj MEDICI10    13))

(13 (TESTS (IS obj SOLID)        14    16))

(16 (TESTS (IS obj LIQUID)       17    25))

(25 (TESTS (IS obj GAS)          26    28))

(28 (TESTS (IS obj ANIMATE)       7     (COMPLAIN 28 25 16 13 9 3)))
```

In this case, the complaint issued at node 28 will be activated if there is some token in the obj register, filled from the OBJECT role of the INGEST driver, which has none of the features BEAST,SOLID, LIQUID,GAS,ANIMATE, and which is furthermore not known to be a MEDICINE. Thus, this complaint implies that one (or more) of the tests must succeed for the INGEST event to make sense.

Responding to complaints

When a complaint is issued, the system looks at the register being tested by the node issuing the complaint, expecting the token contained in that register to be a dummy token. If the token is not a dummy, the inferencer declares that the sentence analyser has delivered a faulty analysis: in such a case, the analyser's design permits it to find different analyses, although this feature is deliberately suppressed in the implemented program.

When the token which has provoked a complaint is indeed a dummy token, this usually reflects a pronoun in the text, and a referent must be found for this. To start the search for a referent, the inferencer runs through each of the indicated tests again, to record the answers given when the dummy token is used. It then has some information it can use to screen potential referents, since any referent for the pronoun must give different answers to at least one of the tests.

It should be apparent that the pronouns HIM/HE/HIS and HER/SHE will never provoke complaints of this kind, since the tokens used to represent these pronouns - DUMMY-MALEn and DUMMY-FEMALEn - have the features MALE and FEMALE respectively, both of which imply the feature HUMAN. They will therefore give the same answer to any test as such tokens as FRED or MARY, and so complaints are of no use in handling this form of pronominal reference. Conversely, the pronoun IT is represented by tokens of the form DUMMY-THINGn, which have the features (ANYTHING NOTHUMAN), and THEY/THEM/THEIR are represented by tokens of the form DUMMY-UNKNOWNSn, having the features (ANYTHING PLURAL NOTFLUID). In each of these case the features are non-specific, permitting the tokens to match almost anything else, but giving negative answers to most feature tests, and hence often provoking complaints.

When a complaint occurs, the environment in which inference is currently taking place, and information relevant to resumption of the processing, are temporarily packed away until the present inference process can profitably be resumed. The environment comprises the inference net, the driver, the set of registers, the name of the register which provoked trouble, the token contained in that register, and the results of all the tests using the dummy token.

Since the conceptual representation delivered by the sentence analyser may contain several conceptualisations, which may correspond to the use of relative clauses, possessives, adjectives etc. in the sentence, suspending one inference path following one complaint allows the inferencer to consider other conceptualisations. Each of these will be used to drive the inference process; and when a complaint is issued, the system proceeds at once with any remaining conceptualisations. Any number of these may in turn provoke complaints, and any that do so will themselves be suspended as above. Finally, when all have been considered by the inference nets, the suspended processes are restarted.

To restart a suspended inference-drawing process, it is necessary to overcome the complaint which caused it to be suspended in the first place, and this could be done in one of two ways. Firstly, it could be assumed that the troublesome token has the feature(s), or other characteristics, which the inference network demands; alternatively, the program could hunt around seeking candidate referents which satisfy the relevant tests, and use these in place of the dummy token which is causing the problem. The second option has been chosen, for two reasons.
(i) In the case of simple stories, such as those which the current program handles, there tend to be only a small number of candidates which satisfy these constraints, because the number of distinct objects mentioned in such stories is very small; it is cheaper to adopt this approach because very few candidates need to be tried. In understanding a novel, however, the first option would be very much cheaper: the number of candidates for "THEY" would typically be enormous, and it would be ridiculous to try out a candidate which was last mentioned some hundred pages ago.
(ii) To implement the first option, the inference nets would have to take explicit account of the fact that newly-generated dummy tokens might be used; for instance, the ATRANS example shown above requires that the actor role has the feature BEAST; if a token, DUMMY-BEAST1 say, were created to cope with a complaint, further tests which determine whether the actor is HUMAN or NON-HUMAN would have to be incorporated. In short, there would be administrative problems if the first option were chosen. (However, it could in principle be the right option.)

The system therefore has to find candidate referents for the dummy tokens which have provoked complaints, and it usually has to do this in an environment where, because the processing was suspended, there are few inferences which have been made: it thus cannot judge these candidates with any reliability, still less rank them in order of likelihood. It nevertheless has to do the best it can: its strategy is to try out the candidates, and see how good an agreement can be obtained with inferences from other sources.

Candidates are found by inspecting the appearances property of the dummy token, leading to the set of propositions in which it appears; the processor then inspects the specifications property of each such proposition, to discover if similar propositions exist elsewhere, and in particular to discover whether the dummy token is one of the tokens which would have to be replaced if these propositions were to be considered identical. When this occurs, the token with which it is paired is a candidate referent: it is known to have compatible features, and there is some reason to believe that these tokens might refer to the same object, since they participate in similar propositions.

Since the search for potential referents is a critical phase in the processing of these dummy tokens, it is essential that some candidate(s) be found, and for this reason all inference nets make implicit inferences, predicating the existence of all tokens. These EXIST propositions provide a fall-back mechanism, since they consist solely of the token and some time reference, and these time references are so loosely specified that they will almost always match one another.

When candidates have been found, the suspended environment is restored. Each candidate in turn is placed in the problematic register, and the relevant tests are reapplied. If a candidate gives a different answer to some test, the inference net is reentered at the appropriate branch from that test; if it gives the same answers as the dummy token itself, the candidate is discarded.

When resuming the inference process in this way, a new proposition is created which asserts that the dummy token and the candidate are identical. This IDENTITY token is placed on the assumptions list, and the normal inference mechanisms will then place this identity token on the <u>assuming</u> property of all subsequent inferences, and will also appropriately maintain the <u>assumption-indicates</u> property of the identity token.

Thus, the groups of inferences which are made when various candidates are assumed can easily be isolated, to facilitate the choice of the best group, and hence the best candidate. There are frequently inferences which are independent of these assumed matches, ie. no matter which candidate is considered, these inferences are made. The groups of inferences resulting from different choices of candidate are then inspected: their specifications indicate other similar propositions, and the system selects the candidate whose resulting inferences show the best agreement (of certainty ratings) with inferences drawn from other sources, provided that this agreement is substantially better than the agreement for any other candidate; if there is little to choose between two candidates, the resolution of this pronoun is deferred for the more general processes described in the next section. In either case, when such a candidate is eventually chosen, the inferences dependent on the identification of the dummy token with other candidates are forgotten.

8.6.2) <u>The resolution of other pronouns</u>

The technique described above for the resolution of pronouns is applicable only in cases where the inference nets complain about IT, THEM or WH-words. It is not, however, an ad-hoc technique for finding referents for these; rather, it solves the problem of how an inference net may proceed when it discovers that the driving conceptualisation contains insufficient information, and has the side-effect of assisting in the resolution of certain pronouns.

Not all pronouns provoke complaints: HIMs and HERs never do so, and there are many cases where THEMs and ITs do not either. Even when complaints are provoked, there are occasions when the inferences made as a result of assuming certain referents do not give substantially different results, in terms of agreement between inferences from several sources. On these occasions, the decision on referents for these pronouns is deferred.

Much pronoun resolution is done when all inference networks have finished, at which time the reference processes have available the maximum information and so can make the most informed decisions. The system attempts to compact its representation of a text as much as possible, and to do so it makes certain assumptions about the referents of pronouns, the answers to questions, the referents of definitely-referred tokens, the identity of time tokens, and the identity of internally-generated dummy tokens. To demonstrate understanding of the text, the system generates English sentences when it resolves pronouns or answers questions, but this generation facility is suppressed when time tokens, or internally-generated dummy tokens, are identified.

The representation of the story is not a structured description of the story, nor is it based upon higher-level structures such as scripts or frames. Instead, it is a distributed set of individual propositions, each of which has inf-links to other propositions, and which may also indicate, via specifications, other similar propositions. To compact such a representation is to reduce the number of discrete propositions, and this is achieved by attempting to merge (*18) similar propositions.

For example, suppose inferences have been made from an input about monkeys and bananas, which result in the creation of the propositions POSS1, POSS3 and POSS5, and also their inverses POSS2, POSS4 and POSS6. The corresponding CD forms and specifications, somewhat simplified, might be as shown:

```
POSS1: ((STATE (STATENAME POSS)           ((POSS3 ((BANANA1 . DUMMY-THING1)
               (THING BANANA1)                     (TIMESPAN1 . TIMESPAN2))))
               (VAL FRED)
               (TIME (NAMED TIMESPAN1))))

POSS3: ((STATE (STATENAME POSS)           ((POSS1 ((DUMMY-THING1 . BANANA1)
               (THING DUMMY-THING1)                (TIMESPAN2 . TIMESPAN1)))
               (VAL FRED)                   (POSS5 ((FRED . DUMMY-MALE1)
               (TIME (NAMED TIMESPAN2))))          (TIMESPAN2 . TIMESPAN3))))

POSS5: ((STATE (STATENAME POSS)           ((POSS3 ((DUMMY-MALE1 . FRED)
               (THING DUMMY-THING1)                (TIMESPAN3 . TIMESPAN2))))
               (VAL DUMMY-MALE1)
               (TIME (NAMED TIMESPAN3))))
```

Note that, as the example shows, a proposition may have several specifications: in this case, if DUMMY-THING1 is replaced by BANANA1 in POSS3, and TIMESPAN2 is replaced by TIMESPAN1, then POSS3 will become identical to POSS1; conversely, if FRED is replaced by DUMMY-MALE1, and TIMESPAN2 is replaced by TIMESPAN3, then POSS3 will become identical to POSS5. For illustrative purposes, I assume that TIMESPAN1 is not compatible with TIMESPAN3, and therefore there is no specification linking POSS1 and POSS5.

In the final stage of inferential processing, all the specifications existing on any proposition are collected together. Using the example, the specification linking POSS1 to POSS3 would be considered relevant to the identification of DUMMY-THING1, whilst both of POSS3's specifications would be relevant to TIMESPAN2.

If the specifications shown above were the only ones that existed, which we unrealistically assume in order to keep the example manageable, the groups of specifications relevant to the identification of various tokens would be:

---

*18 Merging also occurs when two EXIST propositions are inferred which differ only in their time references: in our world, things have one existence only.

For token	specification	appearance	suggestion
TIMESPAN1	(POSS3 ((BANANA1 . DUMMY-THING1) (TIMESPAN1 . TIMESPAN2)))	POSS1	TIMESPAN2
TIMESPAN2	(POSS1 ((DUMMY-THING1 . BANANA1) (TIMESPAN2 . TIMESPAN1)))	POSS3	TIMESPAN1
	(POSS5 ((FRED . DUMMY-MALE1) (TIMESPAN2 . TIMESPAN3)))	POSS3	TIMESPAN3
TIMESPAN3	(POSS3 ((DUMMY-MALE1 . FRED) (TIMESPAN3 . TIMESPAN2)))	POSS5	TIMESPAN2
DUMMY-MALE1	(POSS3 ((DUMMY-MALE1 . FRED) (TIMESPAN3 . TIMESPAN2)))	POSS5	FRED
DUMMY-THING1	(POSS1 ((DUMMY-THING1 . BANANA1) (TIMESPAN2 . TIMESPAN1)))	POSS3	BANANA1

Supposing that the propositions POSS1, POSS3 and POSS5 had certainty ratings of 0.7, 0.4 and -0.5 respectively, this would indicate that matching POSS1 and POSS3 was preferable to matching POSS3 and POSS5. (Real situations, of course, are not this simple.) In determining which identifications to make, and hence which propositions to merge, the system takes account of the agreement of the certainty measures, and also of the number of other identifications which would have to be made for any pair of propositions to be merged. For each dummy token and time token, it constructs a data structure containing the information shown above, split into subgroups depending on the suggested referent; it then computes a measure of how likely each resolution is, and selects the most likely. The resolution is performed, substitution takes place throughout the set of propositions, and any propositions which have become identical are then merged. Also, any specifications which have become invalid as a consequence of the identification process are deleted; this task is complicated in the case where time tokens are identified, because other time tokens may be related to them.

When this process is completed for one token, the process is repeated, possibly resulting in further resolutions.

This procedure applies to many different situations:, since amongst the tokens identified by this process are pronominal tokens, tokens corresponding to wh-words, and tokens created by the analyser to reflect the use of definite reference in the English text.

This process is illustrated in detail in appendix C, where the data structures manipulated in the course of the compaction process for the final sentence of the text below are reproduced in full with appropriate annotations. Given the text

BILL AND JILL WENT TO THE ZOO.
THEY GAVE THE MONKEYS SOME PEANUTS, WHICH THEY ATE.
THEY WENT TO THE RESTAURANT AND DRANK SOME TEA.
JILL TOOK BILL'S MONEY FROM HIM AND GAVE IT TO THE TRAMP WHO WAS TALKING
TO THEM.
WHAT DID SHE GIVE THE TRAMP?

processing the final sentence specifically involves

(a) Choosing JILL, rather than the first-mentioned tramp, as the referent of

"SHE"
(b) Identifying two time tokens, corresponding to the life-times of "She" and "Jill"
(c) Determining that "THE TRAMP" means the same tramp as was mentioned earlier
(d) Choosing the money, rather than the peanuts, the tea, or even the monkeys, as the answer to the question
(e) Merging together all the newly-introduced time tokens.

As the appendix shows, the inference mechanism successfully does this invoking the apparatus described above.

Chapter 9: Demonstration of the inferencer.

In this chapter, the program itself does the talking. Various options have been set to print out a great deal of information about what the inferencer is doing, though much of this information has been removed again for the sake of brevity. For the first two sentences, however, the only modification to the programs output is commentary, preceded again by ";".

The text, as promised, is the same as in the introductory chapter and Chapter 6. This time, the analyser's output is suppressed, and the generator is invoked to express all the inferences dictated by the inference networks.

Virtually all the aspects of the inferencer are demonstrated in this chapter: the results of canonicalisation, the use of inference types, the addition of specifications, state updating, linking to assumptions, complaints and the resumption of inferences afterwards, the use of script-like inference nets, and of course pronoun resolution and question answering.

The "memory" is empty at the start of the run, and so very few matches are found for the earlier sentences. After a while, matches become the rule rather than the exception, and so the printing of "specification" links is omitted after the third sentence: otherwise, I would need about another hundred pages.

```
Pray continue:
BILL AND JILL WENT TO THE ZOO.
[]Now make inferences
                          Created token: PTRANS1    ; These messages are given
                          Attachments: none         ; when a CDform is
                                                      ; "internalised"
                                                      ; "attachments" are the same
                                                      ; as "specifications"

Now try to produce inferences based on
            ((EVENT (ACTOR GROUP£1)
                      (ACT PTRANS)                    ; this structure has been
                      (OBJECT GROUP£1)                ; canonicalised: special
                      (FROM DUMMY-PLACE2)             ; GROUP tokens have been
                      (TO ZOO1)                       ; created to facilitate
                      (TIME (NAMED TIMEPOINT2)        ; reference to the group.
                             (COMPARISON
                                 (BEFORE *NOW*)))))

The sentence is : JILL AND BILL WENT TO A ZOO.
This is called PTRANS1
               =======

Producing inference
                          Created token: LOC1       ; Certain inferences which
                          Attachments: none         ; involve groups are also
                          Created token: LOC3       ; made for the members of
                          Attachments: none         ; those groups. In this case
                          Created token: LOC5       ; LOC3 and LOC5 correspond
                          Attachments: none         ; to JILL and BILL being
                                                      ; "near the place..."
A certain precondition inference is : JILL AND BILL WERE NEAR THE PLACE FROM
WHERE JILL AND BILL WERE GOING TO A ZOO.
                                                This is called LOC1
Linkage to PTRANS1 is precondition
Certainty is 0.89100000
Interest is 0.59400000

Producing inference
                          Created token: EXIST1
                          Attachments: none
                          Created token: EXIST3
                          Attachments: none
                          Created token: EXIST5
                          Attachments: none

A certain implicit inference is : JILL AND BILL EXISTED.
                                                This is called EXIST1
Linkage to PTRANS1 is implicit
Certainty is 0.99000000
Interest is 0.49500000E-1
```

```
Producing inference
                    Created token: LOC7
                    Attachments: none
                    Created token: LOC9
                    Attachments: none
                    Created token: LOC11
                    Attachments: none

A certain resultative inference is : JILL AND BILL ARRIVED AT A ZOO.
                                  This is called LOC7
Linkage to PTRANS1 is resultative
This is update-of LOC1
Certainty is 0.89100000
Interest is 0.59400000

Producing inference
                    Created token: LOC13
                    Attachments: LOC7
                                 (TIMESPAN5 . TIMESPAN4)
                                           ; This is one of the
                                           ; "specification" links; in
                                           ; this case, if the times of
                                           ; being located near the zoo
                                           ; are the same - in fact and
                                           ; in Jill & Bill's hope -
                                           ; then the propositions are
                                           ; identical.
                    Created token: GOAL1
                    Attachments: none

A probable motivation inference is : JILL AND BILL DESIRED THAT JILL AND
BILL ARRIVE AT A ZOO.
                                  This is called GOAL1
Linkage to PTRANS1 is motivation
Certainty is 0.69299999
Interest is 0.39599999

Producing inference
                    Created token: LOC15
                    Attachments: none
                    Created token: GOAL3
                    Attachments: none

An alternative possible motivation inference is : JILL AND BILL DESIRED THAT
JILL AND BILL NOT BE NEAR THE PLACE,FROM WHERE THEY HAD GONE TO A ZOO.
                                  This is called GOAL3
Linkage to PTRANS1 is motivation
Certainty is 0.39599999
Interest is 0.49500000
    ; the "compaction" phase begins now, but in this case there is nothing to
    ; compact. The only link was between LOC7 and LOC13; but since LOC13 was
    ; subordinate to GOAL1, and therefore had no 'certainty' rating, the
    ; program refrains from jumping to conclusions.
```

```
Pray continue:
THEY GAVE THE MONKEYS SOME PEANUTS, WHICH THEY ATE.
[ ]Now make inferences
                          Created token: INGEST1
                          Attachments: none

Now try to produce inferences based on
            ((EVENT (ACTOR DUMMY-UNKNOWNS2)        ; This is one of the two
                    (ACT INGEST)                   ; structures returned by the
                    (OBJECT PEANUTS3)              ; canonicalisation phase for
                    (TIME (NAMED TIMEPOINT12)      ; for this sentence.
                          (COMPARISON
                               (BEFORE *NOW*)))))

The sentence is : SOME ENTITIES ATE SOME PEANUTS.
This is called INGEST1
               =======

Producing inference
                          Created token: EXIST7
                          Attachments: EXIST1
                                     (DUMMY-UNKNOWNS2 . GROUP£1)
                                     (TIMESPAN6 . TIMESPAN3)
                                                 ; This is suggesting the group
                                                 ; as a referent for the "THEY"
                                                 ; but we mustn't be premature

A certain implicit inference is : THE ENTITIES,WHICH HAD EATEN SOME PEANUTS,
EXISTED.
                                   This is called EXIST7
Linkage to INGEST1 is implicit
Certainty is 0.94050000
Interest is 0.98999999E-1

Producing inference
                          Created token: EXIST9
                          Attachments: none
                                                 ; The peanuts have been
                                                 ; DISTINGUISHed from THEY,
                                                 ; so no match can be made.

A certain implicit inference is : SOME PEANUTS EXISTED.
                                   This is called EXIST9
Linkage to INGEST1 is implicit
Certainty is 0.94050000
Interest is 0.98999999E-1
                                                 ; note that only the two EXIST
                                                 ; propositions have been made
                                                 ; before this complaint
____________________________________________________________

A complaint has been issued
These inferences will be continued when DUMMY-UNKNOWNS2 can be resolved
____________________________________________________________
```

```
                      Created token: ATRANS1
                      Attachments: none

Now try to produce inferences based on
       ((EVENT (ACTOR DUMMY-UNKNOWNS1)           ; work now starts on the
               (ACT ATRANS)                      ; other structure given by
               (OBJECT PEANUTS3)                 ; canonicalisation, in the
               (FROM DUMMY-UNKNOWNS1)            ; hope that referents can
               (TO MONKEYS1)                     ; be suggested for this
               (TIME (COMPARISON                 ; awkward pronoun.
                         (AFTER TIMEPOINT2))
                     (NAMED TIMEPOINT9)
                     (COMPARISON
                         (BEFORE *NOW*)))))

The sentence is : SOME ENTITIES GAVE SOME PEANUTS TO SOME MONKEYS.
This is called ATRANS1
               =======

Producing inference
                      Created token: EXIST11
                      Attachments: EXIST7
                                     (MONKEYS1 . DUMMY-UNKNOWNS2)
                                     (TIMESPAN8 . TIMESPAN6)
                                             ; there's a suggestion.
A certain implicit inference is : SOME MONKEYS EXISTED.
                                   This is called EXIST11
Linkage to ATRANS1 is implicit
Certainty is 0.94050000
Interest is 0.98999999E-1

Producing inference
                      Created token: EXIST13
                      Attachments: EXIST7
                                     (DUMMY-UNKNOWNS1 . DUMMY-UNKNOWNS2)
                                     (TIMESPAN9 . TIMESPAN6)
                                             ; there's another one
                                   EXIST1
                                     (DUMMY-UNKNOWNS1 . GROUP£1)
                                     (TIMESPAN9 . TIMESPAN3)

A certain implicit inference is : THE ENTITIES,WHICH HAD GIVEN SOME PEANUTS
TO SOME MONKEYS,EXISTED.
                                   This is called EXIST13
Linkage to ATRANS1 is implicit
Certainty is 0.94050000
Interest is 0.98999999E-1
```

```
Producing inference
                          Created token: EXIST15
                          Attachments: EXIST9
                                              (TIMESPAN10 . TIMESPAN7)

A certain implicit inference is : SOME PEANUTS EXISTED.
                                    This is called EXIST15
Linkage to ATRANS1 is implicit
Certainty is 0.94050000
Interest is 0.98999999E-1

Producing inference
                          Created token: LOC17
                          Attachments: none


A probable precondition inference is : THE ENTITIES,WHICH HAD GIVEN SOME
PEANUTS TO SOME MONKEYS,WERE NEAR THE MONKEYS.
                                    This is called LOC17
Linkage to ATRANS1 is precondition
Certainty is 0.79199999
Interest is 0.39599999

                                                  ; ATRANS is unhappy now.
__________________________________________________________________

A complaint has been issued
These inferences will be continued when DUMMY-UNKNOWNS1 can be resolved
__________________________________________________________________
                          Created token: IDENTITY1
                          Attachments: none
                                                  ; All inference nets have
                                                  ; had a go, so it is time
                                                  ; to try the possible
                                                  ; referents.

Producing inference
                          Created token: POSS1
                          Attachments: none

A certain precondition inference is : IF IT WAS THE MONKEYS WHICH ATE SOME
PEANUTS THEN THE MONKEYS WOULD HAVE THE PEANUTS.
                                    This is called POSS1
Linkage to INGEST1 is precondition
Linking to 'assumptions' (IDENTITY1)              ; the IDENTITY serves as
                                                  ; an assumption for several
                                                  ; inferences, and the
                                                  ; INGEST net is reentered

Certainty is 0.99000000
Interest is 0.49500000

Producing inference
                          Created token: HUNGER1
                          Attachments: none

A probable motivator inference is : IF IT WAS THE MONKEYS WHICH ATE SOME
PEANUTS THEN THE MONKEYS WOULD BE HUNGRY.
```

```
                                  This is called HUNGER1
Linkage to INGEST1 is motivator
Linking to 'assumptions' (IDENTITY1)
Certainty is 0.59400000
Interest is 0.19799999

Producing inference
                    Created token: EXIST17
                    Attachments: EXIST16
                                   (TIMESPAN14 . TIMESPAN10)

A certain resultative inference is : IF IT WAS THE MONKEYS WHICH ATE SOME
PEANUTS THEN THE PEANUTS WOULD CEASE TO EXIST.
                                  This is called EXIST17
                                              ; the program doesn't yet
                                              ; realise that this will be
                                              ; true no matter who ate
                                              ; the peanuts.
Linkage to INGEST1 is resultative
This is update-of EXIST9
Linking to 'assumptions' (IDENTITY1)
Certainty is 0.99000000
Interest is 0.19799999

Producing inference
(DURATION *HOURS*) being ignored
                    Created token: HUNGER3
                    Attachments: none

A possible resultative inference is : IF IT WAS THE MONKEYS WHICH ATE SOME
PEANUTS THEN THE MONKEYS WOULD BE SATED.
                                  This is called HUNGER3
Linkage to INGEST1 is resultative
This is update-of HUNGER1
Linking to 'assumptions' (IDENTITY1)
Certainty is 0.49500000
Interest is 0.19799999
                    Created token: IDENTITY3
                    Attachments: none
                                              ; now try the other
                                              ; plausible candidate.
                                              ; DUMMY-UNKNOWNS1 is no
                                              ; better than the existing
                                              ; token, so is not tried.
```

```
Producing inference
                    Created token: POSS3
                    Attachments: none

A certain precondition inference is : IF IT WAS JILL AND BILL WHO ATE SOME
PEANUTS THEN JILL AND BILL WOULD HAVE THE PEANUTS.
                                             This is called POSS3
Linkage to INGEST1 is precondition
Linking to 'assumptions' (IDENTITY3)
Certainty is 0.99000000
Interest is 0.49500000

Producing inference
                    Created token: HUNGER5
                    Attachments: none


An alternative probable motivator inference is : IF IT WAS JILL AND BILL WHO
ATE SOME PEANUTS THEN JILL AND BILL WOULD BE HUNGRY.
                                             This is called HUNGER5
Linkage to INGEST1 is motivator
Linking to 'assumptions' (IDENTITY3)
Certainty is 0.59400000
Interest is 0.19799999

Producing inference
                    Perfect match: EXIST17
                    Attachments: EXIST16
                                   (TIMESPAN14 . TIMESPAN10)

A certain resultative inference is : IF IT WAS JILL AND BILL WHO ATE SOME
PEANUTS THEN THE PEANUTS WOULD CEASE TO EXIST.
                                             This is called EXIST17
Linkage to INGEST1 is resultative
This is update-of EXIST9
Linking to 'assumptions' (IDENTITY3)
Certainty is 0.99000000
Interest is 0.19799999

Producing inference
                    Created token: HUNGER7
                    Attachments: none

A possible resultative inference is : IF IT WAS JILL AND BILL WHO ATE SOME
PEANUTS THEN JILL AND BILL WOULD BE SATED.
                                             This is called HUNGER7
Linkage to INGEST1 is resultative
This is update-of HUNGER5
Linking to 'assumptions' (IDENTITY3)
Certainty is 0.49500000
Interest is 0.19799999
```

```
                    Created token: IDENTITY5
                    Attachments: IDENTITY3
                                   (DUMMY-UNKNOWNS1 . DUMMY-UNKNOWNS2)
                                                ; now it is time to reenter
                                                ; the ATRANS network, since
                                                ; INGEST has finished.
Producing inference
                    Created token: POSS5
                    Attachments: POSS3
                                   (TIMESPAN20 . TIMESPAN16)

A certain precondition inference is : IF IT WAS JILL AND BILL WHO GAVE SOME
PEANUTS TO SOME MONKEYS THEN JILL AND BILL WOULD HAVE THE PEANUTS.
                                   This is called POSS5
Linkage to ATRANS1 is precondition
Linking to 'assumptions' (IDENTITY5)
Certainty is 0.89100000
Interest is 0.59400000

Producing inference
                    Created token: POSS7
                    Attachments: POSS1
                                   (TIMESPAN21 . TIMESPAN12)

A certain resultative inference is : IF IT WAS JILL AND BILL WHO GAVE SOME
PEANUTS TO SOME MONKEYS THEN THE MONKEYS WOULD ACQUIRE THE PEANUTS.
                                   This is called POSS7
Linkage to ATRANS1 is resultative
This is update-of POSS5
Linking to 'assumptions' (IDENTITY5)
Certainty is 0.89100000
Interest is 0.79199999

Producing inference
                    Created token: POSS9
                    Attachments: POSS3
                                   (TIMESPAN22 . TIMESPAN16)
                    Created token: GOAL5
                    Attachments: none

A possible motivation inference is : IF IT WAS JILL AND BILL WHO GAVE SOME
PEANUTS TO SOME MONKEYS THEN JILL AND BILL WOULD NOT WANT THE PEANUTS.
                                   This is called GOAL5
Linkage to ATRANS1 is motivation
Linking to 'assumptions' (IDENTITY5)
Certainty is 0.19799999
Interest is 0.39599999
```

```
Producing inference
                        Created token: POSS11
                        Attachments: POSS1
                                     (TIMESPAN23 . TIMESPAN12)
                                  POSS7
                                     (TIMESPAN23 . TIMESPAN21)
                        Created token: GOAL7
                        Attachments: none
                        Created token: MLOC1
                        Attachments: none
                                ; this is the inference shown below, but the
                                ; fact that a GROUP is involved causes more
                                ; propositions to be created.
                        Perfect match: POSS11
                        Attachments: POSS1
                                     (TIMESPAN23 . TIMESPAN12)
                                  POSS7
                                     (TIMESPAN23 . TIMESPAN21)
                        Perfect match: GOAL7
                        Attachments: none
                        Created token: MLOC3
                        Attachments: none
                        Perfect match: POSS11
                        Attachments: POSS1
                                     (TIMESPAN23 . TIMESPAN12)
                                  POSS7
                                     (TIMESPAN23 . TIMESPAN21)
                        Perfect match: GOAL7
                        Attachments: none
                        Created token: MLOC5
                        Attachments: none

An alternative possible motivation inference is : IF IT WAS JILL AND BILL
WHO GAVE SOME PEANUTS TO SOME MONKEYS THEN JILL AND BILL WOULD THINK THAT
THE MONKEYS WANTED THE PEANUTS.
                                   This is called MLOC1
Linkage to ATRANS1 is motivation
Linking to 'assumptions' (IDENTITY5)
Certainty is 0.49500000
Interest is 0.39599999

                                          ; so far, inferences have
                                          ; been drawn only from
                                          ; ATRANS; now, the FOOD
                                          ; "function" attached to
                                          ; PEANUTS takes over, by
                                          ; switching to another net.
```

```
Producing inference
                    Created token: JOY1
                    Attachments: none
                    Created token: GOAL9
                    Attachments: none
```

An alternative possible motivation inference is : IF IT WAS JILL AND BILL WHO GAVE SOME PEANUTS TO SOME MONKEYS THEN JILL AND BILL WOULD DESIRE THAT THE MONKEYS BECOME HAPPY.

This is called <u>GOAL9</u>

```
Linkage to ATRANS1 is motivation
Linking to 'assumptions' (IDENTITY5)
Certainty is 0.49500000
Interest is 0.39599999

Producing inference
                    Created token: HUNGER9
                    Attachments: HUNGER1
                                 (TIMESPAN25 . TIMESPAN13)
                    Created token: MLOC7
                    Attachments: none
                    Perfect match: HUNGER9
                    Attachments: HUNGER1
                                 (TIMESPAN25 . TIMESPAN13)
                    Created token: MLOC9
                    Attachments: none
                    Perfect match: HUNGER9
                    Attachments: HUNGER1
                                 (TIMESPAN25 . TIMESPAN13)
                    Created token: MLOC11
                    Attachments: none
```

An alternative probable motivation inference is : JILL AND BILL THOUGHT THAT SOME MONKEYS HAD BEEN HUNGRY.

This is called <u>MLOC7</u>

```
Linkage to ATRANS1 is motivation
Certainty is 0.69299999
Interest is 0.19799999

Producing inference
                    Created token: INGEST3
                    Attachments: INGEST1
                                 (MONKEYS1 . DUMMY-UNKNOWNS2)
                                 (TIMESPAN26 . TIMEPOINT12)
```

A possible follevent inference is : SOME MONKEYS STARTED EATING SOME PEANUTS.

This is called <u>INGEST3</u>

```
; this is of course a very helpful inference in this story. However,
; even if this inference is omitted AD-HAC can still resolve the
; pronoun, by reasoning that the peanuts must have existed while they
; were being given: therefore they were given before they were eaten:
; therefore the monkeys had them when they were eaten
```

```
Linkage to ATRANS1 is follevent
Certainty is 0.29700000
Interest is 0.49500000
      ; now compaction begins again. Some substitutions take place, resulting
      ; in some propositions becoming identical. They are merged, and some
      ; are forgotten.
                              ** ** ** Destroying: EXIST15
                              ** ** ** Destroying: EXIST16
                              ****** forgetting EXIST16
                              ****** forgetting EXIST15
These possible-matches were deleted
(POSS3 . POSS5)                    ; the specifications have to be kept
(POSS5 . POSS3)                    ; up to date when identifications are made


Pronoun resolution : IT WAS JILL AND BILL WHO GAVE SOME PEANUTS TO SOME
MONKEYS.
       ; there was only one possibility for this pronoun


Pronoun resolution : IT WAS THE MONKEYS WHICH ATE SOME PEANUTS.
       ; but there were two possibilities for this.
                              ****** forgetting POSS4
                              ****** forgetting POSS3
                              ****** forgetting HUNGER6
                              ****** forgetting HUNGER5
                              ****** forgetting HUNGER8
                              ****** forgetting HUNGER7
                              ****** forgetting IDENTITY4
                              ****** forgetting IDENTITY3
                     Created token: LOC19
                     Attachments: none
                     Created token: LOC21
                     Attachments: none
                     Created token: EXIST16
                     Attachments: EXIST3
                                    (TIMESPAN9 . TIMESPAN3)
                     Created token: EXIST19
                     Attachments: EXIST5
                                    (TIMESPAN9 . TIMESPAN3)
       ; further inferences are made during compaction, in order to extract
       ; information about members of groups.
** ** ** EXIST7 and EXIST13 can no longer match
** ** ** EXIST7 and EXIST1 can no longer match
                              ** ** ** Destroying: EXIST5
                              ** ** ** Destroying: EXIST1
                              ** ** ** Destroying: EXIST3
                              ** ** ** Destroying: EXIST4
                              ****** forgetting EXIST4
                              ****** forgetting EXIST3
            ; and so on, destroying and forgetting.
```

```
Pray continue:
THEY WENT TO THE RESTAURANT AND DRANK SOME TEA.
[ ]Now make inferences

  ; From this point on, the messages from the matcher have been eliminated .
  ; The reader should have some idea of what is happening by now.
Now try to produce inferences based on
          ((CONJUNCT (FIRST (EVENT (ACTOR DUMMY-UNKNOWNS3)
                                   (ACT PTRANS)
                                   (OBJECT DUMMY-UNKNOWNS3)
                                   (FROM DUMMY-PLACE4)
                                   (TO RESTAURANT1)
                                   (TIME (COMPARISON
                                             (AFTER TIMEPOINT9))
                                         (NAMED TIMEPOINT46)
                                         (COMPARISON
                                             (BEFORE *NOW*)))))
                     (SECOND (EVENT (ACTOR DUMMY-UNKNOWNS3)
                                    (ACT INGEST)
                                    (OBJECT TEA1)
                                    (TIME (COMPARISON
                                              (AFTER TIMEPOINT9))
                                          (NAMED TIMEPOINT49)
                                          (COMPARISON
                                              (BEFORE *NOW*))
                                          (COMPARISON
                                              (AFTER TIMEPOINT46)))))))

The sentence is : SOME ENTITIES MOVED TO A RESTAURANT AND THOSE ENTITIES
DRANK SOME TEA.
This is called CONJUNCT1

A certain precondition inference is : THE ENTITIES,WHICH WERE MOVING TO A
RESTAURANT,WERE NEAR SOMEWHERE,OTHER THAN THE RESTAURANT.
                                    This is called LOC23
Linkage to PTRANS3 is precondition
Certainty is 0.89100000
Interest is 0.59400000

A certain implicit inference is : THE ENTITIES,WHICH HAD MOVED TO A
RESTAURANT,EXISTED.
                                    This is called EXIST8
Linkage to PTRANS3 is implicit
Certainty is 0.99000000
Interest is 0.49500000E-1

A certain resultative inference is : THE ENTITIES,WHICH HAD MOVED TO A
RESTAURANT,ARRIVED AT THE RESTAURANT.
                                    This is called LOC25
Linkage to PTRANS3 is resultative
This is update-of LOC23
Certainty is 0.89100000
Interest is 0.59400000
```

A complaint has been issued
These inferences will be continued when DUMMY-UNKNOWNS3 can be resolved

A certain implicit inference is : THE ENTITIES,WHICH HAD MOVED TO A RESTAURANT,EXISTED.

This is called EXIST6

Linkage to INGEST2 is implicit
Certainty is 0.94050000
Interest is 0.98999999E-1

A certain implicit inference is : SOME TEA EXISTED.

This is called EXIST2

Linkage to INGEST2 is implicit
Certainty is 0.94050000
Interest is 0.98999999E-1

A complaint has been issued
These inferences will be continued when DUMMY-UNKNOWNS3 can be resolved

A probable motivation inference is : IF IT WAS JILL AND BILL WHO MOVED TO A RESTAURANT THEN JILL AND BILL WOULD DESIRE THAT THEY ARRIVE AT THE RESTAURANT.

This is called GOAL11

Linkage to PTRANS3 is motivation
Linking to 'assumptions' (IDENTITY9)
Certainty is 0.69299999
Interest is 0.39599999

An alternative possible motivation inference is : IF IT WAS JILL AND BILL WHO MOVED TO A RESTAURANT THEN JILL AND BILL WOULD DESIRE THAT THEY NOT BE NEAR THE PLACE,WHERE THEY HAD BEEN.

This is called GOAL13

Linkage to PTRANS3 is motivation
Linking to 'assumptions' (IDENTITY9)
Certainty is 0.39599999
Interest is 0.49500000

; now the "GO-TO-RESTAURANT" inference network
; takes over, providing a fair number of situation-
; specific inferences. Unfortunately, the generator
; makes lots of mistakes here: "A FOOD", "SOME
; WAITRESS" for instance.
; Because these are mostly 'follevent' inferences,
; and therefore not anchored to the past tense of
; the story, the generator is obliged to use some
; tense: it expresses such times by using present
; tense, because it cannot "prove" the times to be
; either BEFORE or AFTER *NOW*

```
A probable motivator inference is : JILL AND BILL WERE HUNGRY.
                                    This is called HUNGER11
Linkage to PTRANS3 is motivator
Certainty is 0.69299999
Interest is 0.29700000

A probable follevent inference is : JILL AND BILL ASK SOME WAITRESS FOR A
FOOD.
          ; note that the generator thinks WAITRESS is a plural!
                                    This is called MTRANS1
Linkage to PTRANS3 is follevent
Certainty is 0.79199999
Interest is 0.19799999

A probable follevent inference is : SOME WAITRESS GIVES A FOOD TO JILL AND
BILL.
                                    This is called ATRANS5
Linkage to PTRANS3 is follevent
Certainty is 0.79199999
Interest is 0.19799999

A probable follevent inference is : JILL AND BILL INGEST A FOOD.
                                    This is called INGEST5
Linkage to PTRANS3 is follevent
Certainty is 0.79199999
Interest is 0.19799999

A probable follevent inference is : JILL AND BILL GIVE SOME MONEY TO A
RESTAURANT.
                                    This is called ATRANS7
Linkage to PTRANS3 is follevent
Certainty is 0.79199999
Interest is 0.19799999

A probable follevent inference is : JILL AND BILL LEAVE A RESTAURANT.
                                    This is called PTRANS5
Linkage to PTRANS3 is follevent
Certainty is 0.79199999
Interest is 0.19799999

A certain precondition inference is : IF IT WAS JILL AND BILL WHO MOVED TO A
RESTAURANT THEN JILL AND BILL WOULD HAVE SOME TEA.
                                    This is called POSS2
Linkage to INGEST2 is precondition
Linking to 'assumptions' (IDENTITY9)
Certainty is 0.99000000
Interest is 0.49500000
```

```
A probable motivator inference is : IF IT WAS JILL AND BILL WHO MOVED TO A
RESTAURANT THEN JILL AND BILL WOULD BE THIRSTY.
                                    This is called THIRST1
Linkage to INGEST2 is motivator
Linking to 'assumptions' (IDENTITY9)
Certainty is 0.69299999
Interest is 0.39599999

A certain resultative inference is : IF IT WAS JILL AND BILL WHO MOVED TO A
RESTAURANT THEN SOME TEA WOULD CEASE TO EXIST.
                                    This is called EXIST4
Linkage to INGEST2 is resultative
This is update-of EXIST2
Linking to 'assumptions' (IDENTITY9)
Certainty is 0.99000000
Interest is 0.19799999

A probable resultative inference is : IF IT WAS JILL AND BILL WHO MOVED TO A
RESTAURANT THEN JILL AND BILL WOULD BECOME SLAKED.
                                    This is called THIRST3
Linkage to INGEST2 is resultative
This is update-of THIRST1
Linking to 'assumptions' (IDENTITY9)
Certainty is 0.59400000
Interest is 0.29700000
                      ; the inferences continue now using MONKEYS1 as a
                      ; candidate resolution for the pronoun. The inference
                      ; nets (including the GO-TO-RESTAURANT network) do not
                      ; know any better.
An alternative probable motivation inference is : IF IT WAS THE MONKEYS
WHICH MOVED TO A RESTAURANT THEN THE MONKEYS WOULD DESIRE THAT THEY ARRIVE
AT THE RESTAURANT.
                                    This is called GOAL15
Linkage to PTRANS3 is motivation
Linking to 'assumptions' (IDENTITY11)
Certainty is 0.69299999
Interest is 0.39599999

An alternative possible motivation inference is : IF IT WAS THE MONKEYS
WHICH MOVED TO A RESTAURANT THEN THE MONKEYS WOULD DESIRE THAT THEY NOT BE
NEAR THE PLACE,WHERE THEY HAD BEEN.
                                    This is called GOAL17
Linkage to PTRANS3 is motivation
Linking to 'assumptions' (IDENTITY11)
Certainty is 0.39599999
Interest is 0.49500000

An alternative probable motivator inference is : SOME MONKEYS WERE HUNGRY.
                                    This is called HUNGER13
Linkage to PTRANS3 is motivator
Certainty is 0.69299999
Interest is 0.29700000
```

A probable follevent inference is : SOME MONKEYS ASK SOME WAITRESS FOR A FOOD.
This is called MTRANS3

Linkage to PTRANS3 is follevent
Certainty is 0.79199999
Interest is 0.19799999

A probable follevent inference is : SOME WAITRESS GIVES A FOOD TO SOME MONKEYS.
This is called ATRANS11

Linkage to PTRANS3 is follevent
Certainty is 0.79199999
Interest is 0.19799999

A probable follevent inference is : SOME MONKEYS INGEST A FOOD.
This is called INGEST7

Linkage to PTRANS3 is follevent
Certainty is 0.79199999
Interest is 0.19799999

A probable follevent inference is : SOME MONKEYS GIVE SOME MONEY TO A RESTAURANT.
This is called ATRANS13

Linkage to PTRANS3 is follevent
Certainty is 0.79199999
Interest is 0.19799999

A probable follevent inference is : SOME MONKEYS MOVE FROM A RESTAURANT.
This is called PTRANS7

Linkage to PTRANS3 is follevent
Certainty is 0.79199999
Interest is 0.19799999

A certain precondition inference is : IF IT WAS THE MONKEYS WHICH MOVED TO A RESTAURANT THEN THE MONKEYS WOULD HAVE SOME TEA.
This is called POSS13

Linkage to INGEST2 is precondition
Linking to 'assumptions' (IDENTITY11)
Certainty is 0.99000000
Interest is 0.49500000

An alternative probable motivator inference is : IF IT WAS THE MONKEYS WHICH MOVED TO A RESTAURANT THEN THE MONKEYS WOULD BE THIRSTY.
This is called THIRST5

Linkage to INGEST2 is motivator
Linking to 'assumptions' (IDENTITY11)
Certainty is 0.69299999
Interest is 0.39599999

A certain resultative inference is : IF IT WAS THE MONKEYS WHICH MOVED TO A RESTAURANT THEN SOME TEA WOULD CEASE TO EXIST.

This is called <u>EXIST4</u>

Linkage to INGEST2 is resultative
This is update-of EXIST2
Linking to 'assumptions' (IDENTITY11)
Certainty is 0.99000000
Interest is 0.19799999

A probable resultative inference is : IF IT WAS THE MONKEYS WHICH MOVED TO A RESTAURANT THEN THE MONKEYS WOULD BECOME SLAKED.

This is called <u>THIRST7</u>

Linkage to INGEST2 is resultative
This is update-of THIRST5
Linking to 'assumptions' (IDENTITY11)
Certainty is 0.59400000
Interest is 0.29700000

```
Pronoun resolution : IT WAS JILL AND BILL WHO MOVED TO A RESTAURANT.
                      ; This pronoun resolution has taken place because the
                      ; GROUP is more in focus than the monkeys: there was
                      ; no inference anywhere which really discriminated
                      ; between the two candidates.
                              ****** forgetting POSS14
                              ****** forgetting POSS13
                              ****** forgetting THIRST6
                              ****** forgetting THIRST5
                    ; and so on... the memory is compacted quite extensively
                    ; at this point. The output from compaction has also
                    ; been eliminated.
```

```
Pray continue:
JILL TOOK BILL'S MONEY FROM HIM AND GAVE IT TO THE TRAMP WHO WAS
TALKING TO THEM.
[]Now make inferences

Now try to produce inferences based on
          ((EVENT (ACTOR TRAMP1)
                  (ACT MTRANS)
                  (MOBJECT CONCEPTS1)
                  (FROMCP TRAMP1)
                  (TOCP DUMMY-UNKNOWNS4)
                  (TIME (NAMED TIMEPOINT86)
                        (COMPARISON
                            (BEFORE *NOW*))
                        (COMPARISON
                            (AFTER TIMEPOINT83)))))

The sentence is : A TRAMP STATED SOMETHING.
This is called MTRANS5


A certain resultative inference is : A TRAMP KNEW THAT THE TRAMP HAD STATED
SOMETHING.
                                    This is called MLOC13
Linkage to MTRANS5 is resultative
Certainty is 0.94050000
Interest is 0.19799999

A certain implicit inference is : A TRAMP EXISTED.
                                    This is called EXIST20
Linkage to MTRANS5 is implicit
Certainty is 0.94050000
Interest is 0.98999999E-1

A certain precondition inference is : THE ENTITIES,TO WHICH A TRAMP HAD SAID
SOMETHING,WERE NEAR THE TRAMP.
                                    This is called LOC8
Linkage to MTRANS5 is precondition
Certainty is 0.89100000
Interest is 0.59400000

A certain implicit inference is : THE ENTITIES,TO WHICH A TRAMP HAD SAID
SOMETHING,EXISTED.
                                    This is called EXIST14
Linkage to MTRANS5 is implicit
Certainty is 0.94050000
Interest is 0.98999999E-1
```

```
A probable precondition inference is : A TRAMP PONDERED.
                                    This is called MLOC15
                        ; This is a somewhat pompous way of saying that the
                        ; CONCEPTS1 which the tramp MTRANSed were in the
                        ; tramp's mind - (INCP TRAMP1)
                        ; It is sometimes hard to find English words that
                        ; express what the CDform says. PONDER was the best
                        ; I could do for this.
Linkage to MTRANS5 is precondition
Certainty is 0.79199999
Interest is 0.39599999

A probable motivation inference is : A TRAMP DESIRED THAT THE ENTITIES,TO
WHICH THE TRAMP HAD SAID SOMETHING,PONDER.
                                    This is called GOAL19
Linkage to MTRANS5 is motivation
Certainty is 0.69299999
Interest is 0.89100000

A probable instrumental inference is : A TRAMP SPOKE.
                                    This is called SPEAK1
Linkage to MTRANS5 is instrumental
Certainty is 0.69299999
Interest is 0.59400000
                     Created token: ATTEND1
                     Attachments: none
Asking the memory a question - answers are (ATTEND1 NIL)
      ; Here, for the first time, a MBELIEVE test is being performed.
      ; The token created to represent the question is ATTEND1, and the
      ; matcher has found nothing like that in the memory: hence the NIL.
Now make inferences contingent upon ATTEND1
      ;     Taking one branch from the test.....
The assumption is : THE ENTITIES,TO WHICH A TRAMP WAS SAYING
SOMETHING,LISTENED TO THE TRAMP.

A probable resultative inference is : IF THE ENTITIES,TO WHICH A TRAMP WAS
SAYING SOMETHING,LISTENED TO THE TRAMP THEN THOSE ENTITIES WOULD PONDER.
                                    This is called MLOC19
Linkage to MTRANS5 is resultative
Linking to 'assumptions' (ATTEND1)
Certainty is 0.79199999
Interest is 0.59400000

A certain resultative inference is : IF THE ENTITIES,WHICH HAD BEEN NEAR A
TRAMP,LISTENED TO THE TRAMP THEN THOSE ENTITIES WOULD KNOW THAT HE HAD
STATED SOMETHING.
                                    This is called MLOC21
Linkage to MTRANS5 is resultative
Linking to 'assumptions' (ATTEND1)
Certainty is 0.89100000
Interest is 0.19799999
End of inferences based upon ATTEND1
The assumption is : THE ENTITIES,TO WHICH A TRAMP WAS SAYING
SOMETHING,LISTENED TO THE TRAMP.
Now make inferences contingent upon ATTEND2
      ;     Now taking the other branch from the test.
The assumption is : THE ENTITIES,TO WHICH A TRAMP WAS SAYING
```

SOMETHING,DIDN'T LISTEN TO THE TRAMP.

Created token: MFEEL1
Attachments: none

A possible normative inference is : IF THE ENTITIES,TO WHICH A TRAMP WAS SAYING SOMETHING,DIDN'T LISTEN TO THE TRAMP THEN HE WOULD START TO BE ANGRY WITH THOSE ENTITIES.

This is called MFEEL1

Linkage to MTRANS5 is normative
Linking to 'assumptions' (ATTEND2)
Certainty is 0.39599999
Interest is 0.49500000

A probable normative inference is : IF THE ENTITIES,TO WHICH A TRAMP WAS SAYING SOMETHING,DIDN'T LISTEN TO THE TRAMP THEN HE WOULD STATE SOMETHING.

This is called MTRANS7

```
; Really, this means that he would repeat what he said earlier.
; As noted in Chapter 3, the generator is not making use of the
; inferencer, and so cannot decide to use the word "REPEAT".
```

Linkage to MTRANS5 is normative
Linking to 'assumptions' (ATTEND2)
Certainty is 0.59400000
Interest is 0.29700000
End of inferences based upon ATTEND2
The assumption is : THE ENTITIES,TO WHICH A TRAMP WAS SAYING SOMETHING,DIDN'T LISTEN TO THE TRAMP.

Now try to produce inferences based on

```
((STATE (STATENAME POSS)
        (THING MONEY3)
        (VAL HUMAN-BILL)
        (TIME (NAMED TIMEPOINT83)
              (COMPARISON
                  (BEFORE *NOW*)))))
```

The sentence is : BILL HAD SOME MONEY.
This is called POSS15

A certain implicit inference is : SOME MONEY EXISTED.

This is called EXIST15

Linkage to POSS15 is implicit
Certainty is 0.98010000
Interest is 0.98999999E-1

A certain implicit inference is : BILL EXISTED.

This is called EXIST24

Linkage to POSS15 is implicit
Certainty is 0.98010000
Interest is 0.98999999E-1

A certain exclusion inference is : NOBODY,OTHER THAN BILL,HAD SOME MONEY.
This is called POSS17

Linkage to POSS15 is exclusion
Certainty is 0.94050000
Interest is 0.19799999

A possible causative inference is : BILL TAKES SOME MONEY.
This is called ATRANS15

Linkage to POSS15 is causative
Certainty is 0.39599999
Interest is 0.39599999

An alternative possible causative inference is : SOMEBODY,OTHER THAN BILL,GIVES SOME MONEY TO BILL.
This is called ATRANS17

; This inference causes a lot of trouble later on, when we ask
; "What did she give him?".

Linkage to POSS15 is causative
Certainty is 0.39599999
Interest is 0.39599999

A possible enabledact inference is : BILL CAN GIVE SOME MONEY TO SOMEBODY,OTHER THAN BILL.
This is called ATRANS19

Linkage to POSS15 is enabledact
Certainty is 0.29700000
Interest is 0.39599999

An alternative possible enabledact inference is : SOMEBODY,OTHER THAN BILL,CAN TAKE SOME MONEY FROM BILL.
This is called ATRANS21

Linkage to POSS15 is enabledact
Certainty is 0.19799999
Interest is 0.39599999

Now try to produce inferences based on

```
        ((CONJUNCT (FIRST (EVENT (ACTOR JILL)
                                 (ACT ATRANS)
                                 (OBJECT MONEY3)
                                 (TO JILL)
                                 (FROM DUMMY-MALE1)
                                 (TIME (COMPARISON
                                            (AFTER TIMEPOINT49))
                                       (NAMED TIMEPOINT83)
                                       (COMPARISON
                                            (BEFORE *NOW*)))))
                   (SECOND (EVENT (ACTOR JILL)
                                  (ACT ATRANS)
                                  (OBJECT DUMMY-THING1)
                                  (FROM JILL)
                                  (TO TRAMP1)
                                  (TIME (COMPARISON
                                             (AFTER TIMEPOINT49))
                                        (NAMED TIMEPOINT86)
                                        (COMPARISON
                                             (BEFORE *NOW*))
                                        (COMPARISON
                                             (AFTER TIMEPOINT83)))))))
```

The sentence is : JILL TOOK SOME MONEY AND JILL GAVE SOMETHING TO A TRAMP.
This is called CONJUNCT3

A certain implicit inference is : JILL EXISTED.
This is called EXIST5

Linkage to ATRANS23 is implicit
Certainty is 0.94050000
Interest is 0.98999999E-1

A certain implicit inference is : THE MALE,FROM WHOM JILL HAD TAKEN SOME MONEY,EXISTED.
This is called EXIST22

Linkage to ATRANS23 is implicit
Certainty is 0.94050000
Interest is 0.98999999E-1

A certain implicit inference is : SOME MONEY EXISTED.
This is called EXIST29

Linkage to ATRANS23 is implicit
Certainty is 0.94050000
Interest is 0.98999999E-1

A probable precondition inference is : THE MALE,FROM WHOM JILL HAD TAKEN SOME MONEY,WAS NEAR JILL.
This is called LOC10

Linkage to ATRANS23 is precondition
Certainty is 0.79199999
Interest is 0.39599999

A certain precondition inference is : THE MALE,FROM WHOM JILL WAS TAKING SOME MONEY,HAD THE MONEY.
This is called <u>POSS19</u>
Linkage to ATRANS23 is precondition
Certainty is 0.89100000
Interest is 0.59400000

A certain resultative inference is : JILL ACQUIRED SOME MONEY.
This is called <u>POSS21</u>
Linkage to ATRANS23 is resultative
This is update-of POSS19
Certainty is 0.89100000
Interest is 0.79199999

A certain motivation inference is : JILL WANTED SOME MONEY.
This is called <u>GOAL21</u>
Linkage to ATRANS23 is motivation
Certainty is 0.89100000
Interest is 0.89100000
Asking the memory a question - answers are (MTRANS9 NIL)
Now make inferences contingent upon MTRANS9

The assumption is : THE MALE,WHO HAD SOME MONEY,GAVE JILL PERMISSION TO TAKE THE MONEY.

A certain normative inference is : IF THE MALE,WHO HAD SOME MONEY,GAVE JILL PERMISSION TO TAKE THE MONEY THEN THAT MALE WOULD LIKE JILL.
This is called <u>MFEEL3</u>
Linkage to ATRANS23 is normative
Linking to 'assumptions' (MTRANS9)
Certainty is 0.89100000
Interest is 0.19799999
End of inferences based upon MTRANS9
The assumption is : THE MALE,WHO HAD SOME MONEY,GAVE JILL PERMISSION TO TAKE THE MONEY.
Now make inferences contingent upon MTRANS10

The assumption is : THE MALE,WHO HAD SOME MONEY,DIDN'T GIVE JILL PERMISSION TO TAKE THE MONEY.

A probable resultative inference is : IF THE MALE,WHO HAD SOME MONEY,DIDN'T GIVE JILL PERMISSION TO TAKE THE MONEY THEN JILL WOULD BE REMORSEFUL.
This is called <u>GUILT1</u>
Linkage to ATRANS23 is resultative
Linking to 'assumptions' (MTRANS10)
Certainty is 0.59400000
Interest is 0.49500000

Asking the memory a question - answers are (MLOC23 (MLOC15))
; In this case, the memory has found something which could
; conceivably be an answer to the question - viz. MLOC15
Now make inferences contingent upon MLOC23

The assumption is : THE MALE,FROM WHOM JILL HAD TAKEN SOME MONEY,THOUGHT THAT SOMEBODY,OTHER THAN THAT MALE,HAD TAKEN THE MONEY.

A probable normative inference is : IF THE MALE,FROM WHOM JILL HAD TAKEN SOME MONEY,THOUGHT THAT THE PERSON WHO WAS TAKING THE MONEY HAD TAKEN IT THEN THAT MALE WOULD TELL THE POLICE THAT THAT PERSON TOOK IT.

This is called <u>MTRANS11</u>

Linkage to ATRANS23 is normative
Linking to 'assumptions' (MLOC23 MTRANS10)
Certainty is 0.69299999
Interest is 0.59400000

A certain normative inference is : IF THE MALE,FROM WHOM JILL HAD TAKEN SOME MONEY,THOUGHT THAT THE PERSON WHO WAS TAKING THE MONEY HAD TAKEN IT THEN THAT MALE WOULD WANT IT.

This is called <u>GOAL23</u>

Linkage to ATRANS23 is normative
Linking to 'assumptions' (MLOC23 MTRANS10)
Certainty is 0.89100000
Interest is 0.89100000
Asking the memory a question - answers are (MLOC25 (MLOC15))
Now make inferences contingent upon MLOC25

The assumption is : THE MALE,FROM WHOM JILL HAD TAKEN SOME MONEY,THOUGHT THAT IT BE JILL WHO TOOK THE MONEY.

A probable resultative inference is : IF THE MALE,FROM WHOM JILL HAD TAKEN SOME MONEY,THOUGHT THAT IT BE JILL WHO TOOK THE MONEY THEN THAT MALE WOULD HOPE TO TAKE IT FROM JILL.

This is called <u>MLOC27</u>

Linkage to ATRANS23 is resultative
Linking to 'assumptions' (MLOC25 MLOC23 MTRANS10)
Certainty is 0.79199999
Interest is 0.89100000
End of inferences based upon MLOC25
The assumption is : THE MALE,FROM WHOM JILL HAD TAKEN SOME MONEY,THOUGHT THAT IT BE JILL WHO TOOK THE MONEY.
Now make inferences contingent upon MLOC26

The assumption is : THE MALE,FROM WHOM JILL HAD TAKEN SOME MONEY,DIDN'T THINK THAT IT BE JILL WHO TOOK THE MONEY.

A certain normative inference is : IF THE MALE,FROM WHOM JILL HAD TAKEN SOME MONEY,DIDN'T THINK THAT IT BE JILL TOOK THE MONEY THEN THAT MALE WOULD START TO DESIRE THAT HE KNOW.

This is called <u>GOAL25</u>

Linkage to ATRANS23 is normative
Linking to 'assumptions' (MLOC26 MLOC23 MTRANS10)
Certainty is 0.89100000
Interest is 0.69299999

End of inferences based upon MLOC26
The assumption is : THE MALE,FROM WHOM JILL HAD TAKEN SOME MONEY,DIDN'T THINK THAT IT BE JILL WHO TOOK THE MONEY.
End of inferences based upon MLOC23
The assumption is : THE MALE,FROM WHOM JILL HAD TAKEN SOME MONEY,THOUGHT THAT SOMEBODY,OTHER THAN THAT MALE,HAD TAKEN THE MONEY.
Now make inferences contingent upon MLOC24

The assumption is : THE MALE,FROM WHOM JILL HAD TAKEN SOME MONEY,DIDN'T THINK THAT THE PERSON WHO WAS TAKING THE MONEY HAD TAKEN IT.

A probable resultative inference is : IF THE MALE,FROM WHOM JILL HAD TAKEN SOME MONEY,DIDN'T THINK THAT THE PERSON WHO WAS TAKING THE MONEY HAD TAKEN IT THEN THAT MALE WOULD BE ANGRY BECAUSE HE WANTS IT.
This is called CAUSE9
Linkage to ATRANS23 is resultative
Linking to 'assumptions' (MLOC24 MTRANS10)
Certainty is 0.69299999
Interest is 0.39599999
End of inferences based upon MLOC24
The assumption is : THE MALE,FROM WHOM JILL HAD TAKEN SOME MONEY,DIDN'T THINK THAT THE PERSON WHO WAS TAKING THE MONEY HAD TAKEN IT.
End of inferences based upon MTRANS10
The assumption is : THE MALE,WHO HAD SOME MONEY,DIDN'T GIVE JILL PERMISSION TO TAKE THE MONEY.

A certain implicit inference is : A TRAMP EXISTED.
This is called EXIST31
Linkage to ATRANS25 is implicit
Certainty is 0.94050000
Interest is 0.98999999E-1

A certain implicit inference is : JILL EXISTED.
This is called EXIST33
Linkage to ATRANS25 is implicit
Certainty is 0.94050000
Interest is 0.98999999E-1

A certain implicit inference is : THE THING WHICH JILL HAD GIVEN TO A TRAMP EXISTED.
This is called EXIST35
Linkage to ATRANS25 is implicit
Certainty is 0.94050000
Interest is 0.98999999E-1

A probable precondition inference is : JILL WAS NEAR A TRAMP.
This is called LOC12
Linkage to ATRANS25 is precondition
Certainty is 0.79199999
Interest is 0.39599999

A certain precondition inference is : JILL HAD THE THING WHICH JILL WAS GIVING TO A TRAMP.
This is called POSS29
Linkage to ATRANS25 is precondition
Certainty is 0.89100000
Interest is 0.59400000

```
A certain resultative inference is : A TRAMP ACQUIRED THE THING WHICH JILL
WAS GIVING TO THE TRAMP.
                                    This is called POSS31
Linkage to ATRANS25 is resultative
This is update-of POSS29
Certainty is 0.89100000
Interest is 0.79199999

A possible motivation inference is : JILL DIDN'T WANT THE THING,WHICH JILL
WAS GIVING TO A TRAMP.
                                    This is called GOAL29
Linkage to ATRANS25 is motivation
Certainty is 0.19799999
Interest is 0.39599999

An alternative possible motivation inference is : JILL THOUGHT THAT A TRAMP
WANTED THE THING WHICH JILL WAS GIVING TO THE TRAMP.
                                    This is called MLOC31
Linkage to ATRANS25 is motivation
Certainty is 0.49500000
Interest is 0.39599999

An alternative possible motivation inference is : JILL DESIRED THAT A TRAMP
BECOME HAPPY.
                                    This is called GOAL33
Linkage to ATRANS25 is motivation
Certainty is 0.49500000
Interest is 0.39599999
These possible-matches were deleted
(POSS25 . POSS19)
(POSS19 . POSS25)
(POSS27 . POSS19)
(POSS19 . POSS27)
                              ** ** ** Destroying: EXIST5
                              ** ** ** Destroying: EXIST6
                   ; a fair amount of compaction occurs here, but the trace
                   ; has been removed for space reasons. Then,

Pronoun resolved : IT WAS BILL FROM WHOM JILL TOOK SOME MONEY.

                   ; and a lot more compaction occurs here

Pronoun resolved : IT WAS THE MONEY WHICH JILL GAVE TO A TRAMP.

                   ; and here ...
Pronoun resolved : IT WAS JILL AND BILL TO WHOM A TRAMP SAID SOMETHING.
                   ; and here.
```

```
Pray continue:
WHAT DID SHE GIVE HIM?

                        ; this may look like a trivial question.
                        ; not only does the text give the answer, but
                        ; the program has already demonstrated its
                        ; ability to resolve a pronoun which gave the
                        ; same answer. However,
                        ;  - there are several pronouns to be resolved
                        ;  - the tramp could be either male or female
                        ;  - there is another proposition which can be
                        ;    confused with the one to which this question
                        ;    refers, namely, the possible causative
                        ;    inference suggesting how Bill got his money
                        ;    in the first place.
[]Now make inferences

Now try to produce inferences based on
        ((EVENT (ACTOR DUMMY-FEMALE1)
                (ACT ATRANS)
                (OBJECT ?DUMMY-THING4)
                (FROM DUMMY-FEMALE1)
                (TO DUMMY-MALE2)
                (TIME (NAMED TIMEPOINT161)
                      (COMPARISON
                         (BEFORE *NOW*)))))

The sentence is : A FEMALE GAVE SOMETHING TO A MALE.
This is called ATRANS31
               ========
A certain implicit inference is : THE MALE,WHO HAD RECEIVED
SOMETHING,EXISTED.
                                    This is called EXIST7
Linkage to ATRANS31 is implicit
Certainty is 0.94050000
Interest is 0.98999999E-1

A certain implicit inference is : THE FEMALE,WHO HAD GIVEN SOMETHING TO A
MALE,EXISTED.
                                    This is called EXIST34
Linkage to ATRANS31 is implicit
Certainty is 0.94050000
Interest is 0.98999999E-1

A certain implicit inference is : THE THING WHICH A FEMALE HAD GIVEN TO A
MALE EXISTED.
                                    This is called EXIST21
Linkage to ATRANS31 is implicit
Certainty is 0.94050000
Interest is 0.98999999E-1
```

```
A probable precondition inference is : THE FEMALE,WHO HAD GIVEN SOMETHING TO
A MALE,WAS NEAR THAT MALE.
                                        This is called LOC46
Linkage to ATRANS31 is precondition
Certainty is 0.79199999
Interest is 0.39599999

A certain precondition inference is : THE FEMALE,WHO WAS GIVING SOMETHING TO
A MALE,HAD THAT THING.
                                        This is called POSS22
Linkage to ATRANS31 is precondition
Certainty is 0.89100000
Interest is 0.59400000

A certain resultative inference is : THE MALE,WHO WAS RECEIVING
SOMETHING,ACQUIRED THAT THING.
                                        This is called POSS16
Linkage to ATRANS31 is resultative
This is update-of POSS22
Certainty is 0.89100000
Interest is 0.79199999

A possible motivation inference is : THE FEMALE,WHO WAS GIVING SOMETHING TO
A MALE,DIDN'T WANT THAT THING.
                                        This is called GOAL35
Linkage to ATRANS31 is motivation
Certainty is 0.19799999
Interest is 0.39599999

An alternative possible motivation inference is : THE FEMALE,WHO WAS GIVING
SOMETHING TO A MALE,THOUGHT THAT THAT MALE WANTED THAT THING.
                                        This is called MLOC45
Linkage to ATRANS31 is motivation
Certainty is 0.49500000
Interest is 0.39599999

An alternative possible motivation inference is : THE FEMALE,WHO WAS GIVING
SOMETHING TO A MALE,DESIRED THAT THAT MALE BECOME HAPPY.
                                        This is called GOAL39
Linkage to ATRANS31 is motivation
Certainty is 0.49500000
Interest is 0.39599999
Pronoun resolved : IT WAS JILL WHO GAVE SOMETHING TO A MALE.
                        ; the program does the easy bit first.
                        ; It determines that, no matter who was being
                        ; given and what was being given, Jill was the giver
                        ;
                        ; A lot more compaction ensues, in which the token
                        ; ?DUMMY-THING4 is resolved. Since the program
                        ; realises it is a question token, it saves the
                        ; revelation until the end.
Answering question (1): JILL GAVE SOME MONEY TO A TRAMP.
Pray continue:
(Returning to Lisp...
```

Chapter 10: Some contemporary work

The principles on which AD-HAC is based have been described in the preceding chapters. Many of these principles were derived from earlier work, and that work has been discussed where appropriate: parts of chapters 2 and 3, and all of chapters 4 and 7, have described those earlier systems which were influential in the design of AD-HAC; during its construction, and the subsequent writing of this thesis, there has been other work which addresses many of the same topics. In this chapter I discuss some of this recent work, namely Marcus's PARSIFAL, Boguraev's system for the resolution of linguistic ambiguities, and Small's work on Word Expert Parsing. Particular emphasis is given to similarities and differences between their approaches and mine.

The three other systems described in this chapter are all English analysers, and are all in some respect similar to AD-HAC. There has been other work on inference, but none of which I am aware which bears any great likeness to AD-HAC: therefore that work is not described here.

10.1) Marcus

PARSIFAL [Marcus 1980] is a syntactic parser which operates strictly deterministically, in that it does not simulate a nondeterministic machine. It does however have a limited look-ahead ability, using a buffer of fixed length which may hold words or syntactic constituents, and this is the basis of its similarity to AD-HAC, which works on a bounded number of partial analyses.

The structures built by PARSIFAL are _annotated surface structures_ with _traces_ indicating the underlying positions of "shifted" NPs. One of the claims made is that several currently proposed constraints of universal grammar (Subjacency, the Specified Subject Constraint, and Ross's Complex NP Constraint) closely resemble constraints forced on the grammar rules by the structure of the parser; this structure is in turn motivated by the "Determinism Hypothesis", which states that

> There is enough information in the structure of natural language in general, and English in particular, to allow left-to-right deterministic parsing of those sentences _which a native speaker can analyze without conscious effort_ (p204)

The grammar rules used by PARSIFAL are constrained in several ways, both by the structure of the parser and by the restricted set of actions available for use in the rules. The parser has three basic data structures:

(1) A _buffer_ of five cells, each cell being able to hold either a word or a syntactic constituent. The primitives for manipulating the buffer allow the insertion and removal of items in particular cells, and shuffle the cells along when items are inserted or removed in such a way that the buffer remains compacted at all times.
(2) A _stack_ of syntax nodes, of which the top node and the _current cyclic node_ only are accessible to grammar rules.
(3) A _stack_ of offsets into the buffer, whose top item indicates the start of a "window" of three cells in the buffer.

The grammar rules are able to inspect the two accesible syntax nodes in the stack, and up to three contiguous buffer cells: they are able to push nodes onto the stack from the buffer, pop nodes from the stack into the buffer, insert specific words in the buffer, and attach a constituent held in the buffer to one of the nodes on the stack. These last attachments, however, cannot be broken once made: and this is one of the central restrictions on the grammar which enforces its deterministic action.

Besides the grammar rules which manipulate the stack and the buffer for the purpose of building syntactic structures, there are two other sorts of rule: diagnostic rules, which handle local syntactic ambiguities; and attention shifting rules which initiate the building of certain constituent structures by moving the "window" of three cells across the buffer of five.

The diagnostic rules cope with situations where the same terminal substring could be accounted for by two different derivations; they are specially designed for situations in which more than one of the grammar rules would apply and would build some structure. These diagnostic rules are constrained, like the normal rules, to inspect only three cells in the buffer and the two accessible nodes on the stack, and this constraint appears to render the parser as a whole susceptible to "garden paths", lending further credibility to the parser as a model of human syntactic analysis.

The attention shifting rules are provided to enable the parser to build complete nodes in the buffer for complex constituents which are reliably signalled by a single word. Thus determiners, for instance, reliably signal the beginning of noun phrases, and act as the signal for attention shifting rules to come into play. These rules act by pushing a new value onto the stack of offsets - (3) above - which corresponds to the position in the buffer of the word which caused the rule to fire. Subsequent rules will then be able to access that buffer cell and the next two, until some rule causes the offset stack to be reset.

Attention-shifting rules may in fact leave the window in the same place, since the item which triggers a rule may already be at the start of the buffer. Marcus notes that, in his grammar, there is never any need to have more than two nested attention shifts which do move the window; and that when there are two, neither of them needs to shift it by more than one place. Therefore, he argues, the total size of the buffer need only be five cells long: the grammar rules may inspect three cells at a time, and the offset can be limited to two. He notes that

> Furthermore, five happens to fall nicely within the range of seven plus or minus two, thus leaving open the possibility that the buffer might be equated with the psychologist's notion of short term memory. (p199)

This observation is markedly similar to, though more cautious than, my remark at the end of section 5.2.3, which is repeated here for reference.

> The second difference [from a simplified model] has, I believe, important implications concerning the relation between memory and linguistic faculties. This is the observation that, despite the apparently non-deterministic operation of the [AD-HAC] analyser, the depth of its stack of theories need only be small; the size of stack needed compares with the supposed size of human short-term memory. Specifically, limiting the size of

this stack to seven theories, and simply discarding any theories which fall off the end, actually improves the performance of the analyser for complex sentences, even elaborately embedded ones.

It should be noted that PARSIFAL is intended only to produce syntactic analyses; the annotated surface structures which it builds are supposed to be amenable to semantic analysis. Marcus does point out that semantic judgements are occasionally necessary in order to determine the correct syntactic structure, and that these semantic judgements cannot be conducted in a strictly deterministic fashion. This, I believe, leads to a serious dilemma for the notion of deterministic parsing as a whole, as I argue below. In passing, however, it is interesting to note that Marcus suggests that a language analyser needs to incorporate both syntactic and semantic biases, which must measure "degree of goodness" and must be susceptible to comparison. AD-HAC's "preference" mechanism does precisely this.

The dilemma

Marcus asserts

(1) that syntactic processing is a real phenomenon in human language use, because the availability of annotated surface structures facilitates the later semantic analysis
(2) that syntactic processing may proceed strictly deterministically, and in particular that syntactic structures once built cannot be deleted
(3) that certain semantic judgements are necessary in order to select between competing hypothesised syntactic structures
(4) that these semantic judgements cannot be deterministic.

If (3) and (4) are true, then whatever mechanism performs the semantic judgements must construct alternative semantic structures in order to make a selection; (1) implies that the input to this mechanism takes the form of annotated surface structures. Therefore, one might suppose that the semantic component tries out the alternatives by completing the syntactic characterisation in the various different ways: but (2) says that this is not possible, because once a link has been created it cannot be destroyed. On the other hand, if one supposes that the semantic component tries out the alternatives in some different way, one must also assume that there is some other mechanism available to the semantic component, which permits it to proceed without the syntactic characterisation: which throws doubt on (1).

10.2) Boguraev

[Boguraev 1980] describes an English analyser whose principal aim is to resolve the various forms of ambiguity which may exist in single sentences: the program deals with lexical, structural and transformational ambiguities, but does not yet handle referential ambiguity. The analyser constructs dependency structures which show both the syntactic structure(s) of plausible readings of a sentence, and the semantic relationships between constituents and word senses. These dependency structures are then passed to an English generator which produces "dynamic paraphrases" of each plausible reading; part of the generation process - the production of a sentence from a syntax net - is identical to that in AD-HAC.

The senses of words in Boguraev's system are represented in terms of a set of primitives, a set which is a development of Wilks's primitive terms. The analysis process uses an ATN to perform a syntactic analysis, with calls to semantic routines at strategic points to ensure that the constituents found by the ATN are semantically coherent: this approach considerably reduces the nondeterminism inherent in the ATN model of syntactic analysis.

The system is equipped with a vocabulary of about four hundred words, most of which have more than one sense, and many of which belong to more than one part of speech. The grammar covers a fair range of English constructions, and so sentences are easily constructed for which there are several syntactically valid readings. Boguraev's system performs impressively, giving for almost all sentences only that interpretation which is right (in the sense of being the interpretation assigned by people); and yet giving all plausible readings in the case of genuine ambiguity.

The semantic routines are called from within the ATN grammar whenever a complete constituent has been found, for instance a clause or a noun phrase; they have access to the ATN registers, and generally prune the list of possible interpretations by using all the information available at the time they are called. In many cases, it is not possible to select just one interpretation for a constituent, and so the set of possible readings is passed up to the next level of computation; in particular, the noun phrase specialist cannot usually select between different interpretations of a noun, though it can prune the interpretations in cases where polysemous adjectives and nouns interact. For example, given a noun phrase such as "THE GREEN CROOK", where there are three senses of "GREEN" and two of "CROOK", the noun phrase semantic specialist can determine that the only reasonable interpretations correspond to "THE NOVICE CRIMINAL" and "THE GREEN-COLOURED SHEEP-BEATER".

The largest part of the disambiguation process occurs in the clause-level semantic specialist. This exploits both syntactic and semantic information to select a <u>contextual verb frame</u>. These verb frames encode both the meanings of senses of verbs (in terms of the Wilksian primitives), and their expected syntactic framework; particular emphasis is placed on handling prepositional phrases, whether optional or obligatory. AD-HAC's verb definitions provide equivalent information in the verb definitions, by defining "requests" which seek elements of the syntactic environment: Boguraev contrasts this approach with the "passive" approach adopted in his system; his contextual verb frames are static objects, which are manipulated by a uniform process which inspects the surrounding syntactic (and semantic) environment.

The treatment of adjectives in Boguraev's system is very similar to the treatment of verbs; abstract nouns also are handled in like manner. The problems of interpreting abstract nouns and polysemous adjectives, it will be recalled, are not handled in AD-HAC.

However, AD-HAC does score over Boguraev's system in certain respects. Conjunctions, for example, are quite easily handled in the AD-HAC framework, whereas they are not handled at all in Boguraev's system at present. For the purposes of story understanding, this would not be too serious a drawback, since simple stories can usually be rewritten in a way that avoids the use of conjunctions. More seriously, Boguraev's treatment of tense is woefully superficial: the dependency structures merely mark the tense used in the English clause, without even attempting to derive the temporal relationships between major clauses and embedded clauses in the same sentence. The work on the inference component of AD-HAC has revealed that a good representation of time is of critical importance; for these dependency structures to be useful for inference would require that the inference mechanism was able to construct for itself the time relationships between the various clauses; but the mapping from tense to time is strongly language dependent, and so should be performed by the analyser.

The use of semantic specialists to cut down the number of alternative semantic structures passed up to higher levels has one fatal flaw: it is too easy for the analyser to reject altogether a metaphor. Boguraev reports that his current system rejects the sentence

"Ships proudly ploughing the sea are a magnificent sight".

The semantic specialists merrily throw away all the interpretations of 'ships being proud' and 'ships ploughing' and 'the sea being ploughed', telling the ATN grammar to find another syntactic analysis; the ATN cannot do this, of course, but nor can it return to previously rejected hypotheses: consequently the program fails to build any analysis of the sentence.

Problems like this are averted in AD-HAC, because alternative analyses are carried along in parallel; if there is only one branch to be taken, it is taken despite apparent semantic anomalies. Boguraev's system tries to avoid parallelism, and consequently has too rigid a notion of failure.

10.3) Small

The Word Expert Parser (WEP) [Small, 1980] views individual words as active agents which cooperate to determine the meaning of a sentence; each word has a large number of senses, and the disambiguation of each word contributes to the derivation of a meaning for a sentence.

The theory underlying the WEP is embodied in the formalisms in which words are defined: these are the Lexical Interaction Language and the Sense Discrimination Language. The definitions of words are expressed as coroutines, which send messages to and fro, and gradually refine the concepts to which the words refer.

The representation into which sentences are parsed by WEP has the primitives of CD as its basis, and so WEP may be viewed as a conceptual parser. In WEP however, the representation is really in terms of a much wider range of concepts, arranged in a tangled hierarchy: a concept like C£WALK-DOG is a specialisation of C£WALK, which is in turn a specialised C£PTRANS. Small justifies the use of such specialised concepts in two ways:

1) Instead of representing once in the memory that walking a dog consists of two associated walking actions, and that a walking action consists of moving ones legs in such and such a manner, this information must [in CD] be copied for each representation of the complex action. Each time a conceptual parser reads a sentence about walking, it must construct a relation on changing location, moving body parts, and so forth. (p77)

2) The conceptual notion at the heart of the action described in the example sentence ["Rick walked his dog"] is that of walking a dog. Most people know certain things about this conceptual action, including that (a) the dog often wears a collar, (b) a person holds a leash attached to this collar, and (c) the dog and the person have one particular interest of the dog in mind during the walk. Just as certain inferences derive directly from the primitive ACTs of conceptual dependency, so can inferences be drawn from more specific higher-level actions that compactly

represent more complex or refined conceptual notions. (p78-79)

This approach to the representation of meaning parallels developments of CD carried out in recent years by the Yale A.I. group; their CD representations now may make direct reference to the names of scripts, and give the bindings of certain script variables in the same way that the primitive-based representations give bindings for the conceptual cases of primitive ACTs.

The WEP is intended to function in conjunction with a memory/inference mechanism, and the nature of the expected interactions is one of the most elegant features of the WEP model. The projected interactions are requests to the inferencer to select the most appropriate of a number of views of a concept: the inferencer is expected to reply with the conceptually closest view, and its reply assists the process of sense discrimination. For example, when the parser sees the word "THROW" used as a verb, it initiates a sequence of actions which pick up the direct object of the verb; it then asks the inference mechanism whether this object is most naturally viewed as:

C£ANYTHING
C£MEAL
C£GARBAGE
C£ABSOBJ
C£SMALL-PHYSOBJ
C£PERSON
C£CONTEST
C£PARTY

(These concepts have usually mnemonic names; C£ABSOBJ is "abstract object")
The inference mechanism selects the closest concept, and this selection is then used to guide the discrimination of the sense of THROW.

Though the necessary inferential activities have not been programmed, the parser's dependence on the view mechanism is one of the most striking and elegant features of the WEP model; the ability to incorporate judgements like this in the parser seems to stem directly from the hierarchical nature of the concepts used to represent meaning, and consequently it is difficult to see how such a mechanism could be retro-fitted into AD-HAC's analyser. Furthermore, there are other mechanisms in WEP, for instance the ability to search the discourse context and the permanent real-world memory, which give it some very useful abilities; again, it is hard to envisage modifications of AD-HAC along these lines.

Word Expert Parsing requires the construction of an expert for each word. These experts are in general extremely complex objects: they comprise a number of coroutine entry points; each coroutine is a network containing test and action nodes, somewhat reminiscent of AD-HAC's inference nets. When a sentence is analysed, the experts for the words in the sentence are invoked. The processing that takes place is then entirely in the hands of these experts: they send messages to one another, posting restart demons to await the answers to questions, and also posting timeout demons to handle cases where no answer to a question is forthcoming from another expert; these two kinds of demons allow control within an expert to switch to other coroutine entry points within the same network, or to other points in the same coroutine. The coroutines model a distributed processing scheme, which is deterministic, but is certainly not left-to-right.

Experts, as mentioned, are very complex. This complexity is necessary to permit the discrimination of the many senses of words; the 24 senses available for the word THROW, for instance, are listed below.

```
C£L£THROW                   ; a "concept" for the lexical item "THROW"
C£ORGANISATION-THROW
C£PERSON-THROW
C£HORSE-THROW
C£MACHINE-THROW
C£THROW-GARBAGE-OBJECT
C£THROW-SMALL-PHYSOBJ
C£LOSE-PURPOSELY
C£THROW-PARTY
C£HORSE-THROW
C£HORSE-THROW-SHOE
C£HORSE-THROW-RIDER
C£L£THROWIN                 ; a "concept" for "THROW IN"
C£L£THROWOUT                ; a "concept" for "THROW OUT"
C£DISCHARGE-INTENSELY
C£THROW-AWAY-GARBAGE
C£THROW-OUT-OF-COURT
C£REMOVE-FROM-LOCATION
C£THROW-OBJ-TO-LOC
C£INTERJECT
C£ENGAGE-MACHINE
C£INCLUDE-IN-DEAL
C£GIVE-TO-CHARITY
C£GIVE-UP
```

When a representation is built by WEP, more than one of these concepts will be used to describe the throwing activity described by the sentence, and so these 24 senses are not mutually exclusive. Indeed, some of them - those starting with C£L£ - are concepts which relate simply to surface lexical items, and are not of the same type as the others.

Because each expert is so complex, there are only a few which have been implemented: including experts for some inflectional morphemes and one for a full stop, there are only 40 experts, giving a vocabulary of maybe 35 actual words. One of the problems with the word expert approach is that each expert must be constructed from scratch (and may involve modification of existing experts), there being no mechanism to capture generalisations easily. There seems to be no notion even of nouns, adjectives and verbs, except implicitly in the signals sent between experts for these words and others.

Even given the basic premiss underlying WEP, that words are highly idiosyncratic in their interactions with other words, this lack of generalisation seems unreasonable, simply because these basic syntactic notions provide an elementary and intuitively correct level of description of language. The WEP approach is yet young; perhaps these and other generalisations will eventually find a place in the overall model.

Chapter 11: Conclusions

In the preceding pages, I have described the principles on which AD-HAC is based, and related it to previous and contemporary work in natural language processing. It is appropriate here to consider the successes and failures of the AD-HAC system: its successes illustrate the feasibility of the general approach adopted, and raise a number of interesting theoretical issues; and while some of its failures may be dismissed as mere faults of implementation, others raise more serious questions about some of the fundamental assumptions on which the system is based, and so point out possible directions for future research.

11.1) The separation of analysis and inference

One of the simplifying assumptions in AD-HAC is that language analysis can be largely separated from inference and memory. This is a common partitioning of the overall problem of language comprehension, which is adopted because analysis and inference/memory are both hideously complex activities: in order to conduct any effective research into these areas, it is necessary (as a first approximation, at least) to study each one in relative isolation.

It has been argued [DeJong, 1979] that this separation is false: that memory and inference are inextricably bound up with language analysis. There can be little doubt that this is correct, and that no computer program will be able to understand properly unless it allows for complex interplay between the various processes and structures. However, DeJong makes these observations in introducing FRUMP, a program which does integrate its analysis with its memory processes, and which is consequently forced simply to ignore much of the text it reads: because the analysis process is so strongly geared towards finding text to support its conjectures, it cannot do anything at all with "extraneous" material. This is not a realistic approach to language comprehension either. Research is clearly needed to find a satisfactory way in which the major components of the system can interact, and one approach to this is to find what sort of information would have to be communicated between the components for satisfactory processing. This is most easily done by pretending that the problem is decomposable, and this is the reason for AD-HAC's separation of analyser and inferencer.

11.2) The analyser

11.2.1) Summary of analysis

The analyser is interesting for several reasons. It clearly exploits the syntactic regularities of English, but at no point does it build a syntactic structure: the organisation of its syntactic and semantic processes integrates conceptual structure building with syntactic recognition. The clause-level request macros make it very easy to analyse standard constructions, and provide easily-overridden defaults for such things as the interpretation of tensing patterns.

Four features of the analyser have proved particularly valuable:
(1) the use of a purely syntactic ATN to locate constituent phrases means that part-of-speech ambiguity is usually eliminated by syntactic context alone, and means also that most function words do not need to bear requests;

(2) the embedding of the request mechanism into a preference-directed nondeterministic framework alleviates many of the problems of control which afflicted Riesbeck's system, and simplifies the handling of ambiguity in all its forms;
(3) the identification and cyclic application of five distinct classes of requests, corresponding to five kinds of activity within the parser, further simplifies the control of the analysis process, especially when the idiosyncracies of particular words (for example, conjunctions) necessitate operations on the existing requests.
(4) the addition of a KEEP predicate to requests removes the need for complex interactions among requests, and makes it possible to treat the main predicate strictly as a predicate, and the actions strictly as actions.

11.2.2) Problems for further research in analysis

There are some fairly minor problems with the analyser. As noted in section 5.4, the analyser does not handle reflexive pronouns, participial premodifiers, abstract nouns, or "how" questions; these are areas to which I believe AD-HAC could be extended without radical alteration. Similarly, the analyser's treatment of adjectives is inflexible and simple-minded, and further work is needed on this.

The language understood by the analyser is in some cases restricted because the representation language provides no suitable constructs: for this reason, the analyser cannot handle disjunction, nor can it handle adverbial modifications. The omission of disjunction was an oversight, and the extension of the analyser to cope with disjunction, especially in questions, should prove interesting. Adverbs are an extremely fruitful area for further research: the meaning of adverbs is very hard to determine, and decisions on how adverbs are to be represented must precede decisions on how an analyser is to cope with them.

The possibility that 'CONCEPTS' might need modification was also overlooked when the representation language was developed, and this surprising omission leaves the analyser with no way to represent the notions of
"prediction", which should be "(CONCEPTS (TIME .....))"
"lies", which should be, say, "(CONCEPTS (VERITY ...))"
"a story about Fred", perhaps "(CONCEPTS (ABOUT FRED))"
Note that "ABOUT" and "VERITY" - special purpose roles for just these situations - would need to be treated in a special manner by the inferencer: (VERITY FALSE) would have to specify conceptualisations which were not believed, while (ABOUT FRED) would restrict the concepts to match conceptualisations in which FRED played a role. These are not really problematic for the analyser; rather, the representation language lacks certain facilities.

There are however two serious problems with the analyser, problems which perhaps indicate that the very structure of the analyser is at fault. Firstly, as discussed in 5.4, the analyser suffers from the garden-path syndrome when presented with sentences like

"The horse raced past the barn fell"
"The boat floated down the river sank"
"The cars made in Japan are small"

The first two of these sentences are genuine garden paths; the third is not. Thus, insofar as AD-HAC's analyser does make predictions about garden paths, these predictions are incorrect.

Secondly, though the use of a purely syntactic ATN to locate constituent phrases has many advantages as illustrated above, the validity of constituent-level analysis is thrown into doubt by a phenomenon exemplified by the phrase

"the man in the corner's wife" (Example from Henry Thompson)

AD-HAC gives this a bizarre interpretation: the corner is endowed with a wife, and the man is understood to be in her (whatever that means).

This phrase violates two of the assumptions on which AD-HAC is based: that phrases of the form "noun1's noun2" represent some (loosely possessive) relation between "noun1" and "noun2"; and that phrases of the form "noun-phrase prep-phrase" have "noun-phrase" as their head. This latter assumption is particularly central in AD-HAC, because the request mechanism applies feature tests to the first noun phrase in order to apply its preferences; if the real head of a phrase of the above form has critically different features, this test would lead the analyser astray: if the phrase were embedded in a complex syntactic environment, this misguided test might render the parser unable to derive any analysis at all.

Nor is it the case that the features of the first head noun and the real head noun must be similar, since examples where the features are critically different can be easily constructed: eg.

"I twisted the man in the corner's arm"

Notwithstanding the problems mentioned above, the analyser used in AD-HAC has proved effective, versatile, and easily extensible. Its present vocabulary is of the order of 350 words (listed in appendix B), and continues to grow. Perhaps the most unexpected attribute of the analyser is that it has provided endless innocent enjoyment.

11.3) <u>The interface between analyser and inferencer</u>

The traffic between the analyser and the inferencer is, at present, one way only. The analysis of a sentence cannot exploit extra-sentential context, even at the level of preferring certain senses of individual words. In fact, the analyser's use of features in its preference tests constitutes its only inferential ability; and this is an extremely primitive form of inference.

One of the consequences of this lack of feedback from the inferencer is that the analyser cannot find probable referents for pronouns. This in turn means that it cannot use knowledge of the pronoun's referent even for its feature-preference tests. This is an unsatisfactory state of affairs, but is compounded because the analyser should not enforce its expectations on the inferencer, in case metaphor is involved.

Out of context, the word "THEM" in "HE ATE THEM" might be presumed to refer to some objects with the feature 'SOLID' (which encompasses BEAST and PLANT, as shown in the section 2.3.6); but if the analyser were to note that the feature SOLID was expected, and push this feature on the pronominal token to assist the inferencer in finding the referent, the effects would be unfortunate in:

"Fred regretted his earlier words. He ate them."

A forced example, no doubt, but comprehensible; an analyser should not enforce its expectations on an inference mechanism. Clearly, for this example, the analyser would need to know the referent of the pronoun, and this is beyond AD-HAC's abilities.

The analyser yields, in addition to the conceptual structure it has built, a few lists of interesting items: pronouns, definite references, indefinite references, "proper name" tokens, and "query" tokens. These are collected by the analyser as it proceeds through its various paths of analysis, and the set finally produced corresponds to the set used in the conceptual structure. This mechanism could probably be extended to permit the analyser to specify the features it expected for pronouns, thereby giving the inferencer information available during analysis, but not enforcing the analyser's assumptions. Similarly, the problem of reflexive pronouns and pronominal coreference constraints might be tackled in this fashion.

In AD-HAC, however, the representation of features is distinct from the representation of other conceptual information; and this discrimination is an obstacle to the implementation of such a scheme. In retrospect, it seems that the representation of all kinds of information should be as uniform as possible; the use of property lists of Lisp atoms to store crucial information about tokens now seems indefensible when all other information is stored in a different manner.

Features remain a useful device, though their implementation should be reorganised. However, the use of features is a long way from adequate for representing the meaning of nouns: the use of "functions" attached to nouns, and associated with inference networks which can be activated in appropriate contexts, permits much more knowledge about objects to be deployed; still there are desirable forms of knowledge which are unavailable to the inferencer.

In particular, the inferencer needs the equivalent of a thesaurus. To some extent, the feature-prototype dummy tokens (DUMMY-HUMAN, DUMMY-SOLID, etc.) perform this function; so does the superprimitive classification described in sections 2.3.4 and 8.3.2. Neither of these mechanisms permits the inferencer to realise that "nuts" may refer to "peanuts", however, because the classifications represented by the feature set and the superprimitives are too coarse-grained. The answer to the problem categorically does not lie in introducing more features; some representation of the meanings of nouns, similar to that developed by Wilks, could perhaps perform the required functions. The representation of the meanings of nouns is an urgent topic for future investigation.

11.4) The inferencer

11.4.1) Summary of inference

AD-HAC's inferences are for the most part organised around the conceptual primitives, and specify low-level preconditions, results, motivations and so on; the inferencer adopts a unified approach to the problems of pronoun resolution and question answering, and the low-level inferences alone usually prove sufficient for this task.

The inference process is directed by a set of networks which produce the actual inferences and which can interrogate the existing context. The low-level primitive-oriented networks can also invoke higher-level networks which are associated with the 'functions' of objects; these higher networks encode script-like knowledge. The low- and high-level networks play different roles in the system, but are expressed in the same form: AD-HAC successfully integrates high-level situation-specific knowledge with low-level primitive-oriented knowledge.

This is achieved by casting the inference processes in the form of discrimination networks, where some of the nodes in the net perform tests, and some execute actions. The actions performed involve the construction of conceptual patterns using elements of existing conceptualisations in prespecified templates, thus producing inferences. The inferences fall into natural classes, and these classes provide an interesting and simple relationship between the inferences made from affirmative and negative sentences.

The inferences produced are equipped with extensive cross-references, which are exploited to compact the memory by merging similar inferences drawn from disparate sources: this compaction process has as a natural side effect the identification of pronoun referents and of the answers to WH-questions, leading to a unified approach to these two tasks.

Early consideration of the likely problems of inference led to the adoption of a modified form of Conceptual Dependency as the representation language. The most significant modification of CD has been the use of <u>focus-paths</u> to subordinate one conceptualisation to another. This device enables all information to be represented in a perspicuous and perfectly uniform fashion. The introduction of <u>superprimitives</u> also permits the specification of whole classes of closely related messages, overcoming an awkward aspect of primitive-based representation schemes.

11.4.2) <u>Problems for further research in inference</u>

The inferencer is equipped with several inference nets, but there remains further work to be done. There remain several of the conceptual primitives for which inference networks have not yet been written; and only a very small number of script-like networks exist at present. It remains to be seen whether any problems arise in attempting to encode plan-like knowledge in the form of inference networks, as there currently are no networks of this sort. It should be noted, however, that some provision has been made for the incorporation of plans, in the form of the defined inference types "motivation" and "plannedact".

Amongst the conceptual primitives for which no inference network exists is QUANTIFY: consequently, the inferencer does nothing very useful with universal quantifiers, and cannot draw the obvious conclusions from syllogisms. It appears, however, that creating an inference network for QUANTIFY would be only a small step towards extending the inferencer to handle syllogisms. One approach towards handling the larger problem is to create inference networks dynamically: for a statement such as

"ALL FAT PEOPLE ARE HAPPY"

an inference net could be constructed, on the spot, to perform the necessary tests and make the appropriate inferences. There is no conceptual problem there, but a fair degree of modification to the existing program would be required in order to implement this approach.

Some of the apparent problems of inference are really problems of representation. One of these problems is the representation of different types of causality: the present program merely uses CAUSE to relate an ANTECEDENT and a RESULT; but it appears that there are many different types of causality, and it further seems that the existing classification of inference types corresponds closely to the perceived types of causal relationship. It would therefore be useful to incorporate this classification into the representation language itself.

Another problem is that, in some places, the representation language enforces too high a degree of precision in expression. An example of this occurs in the inferences drawn by AD-HAC from an MTRANS between two different people: one "precondition" inference states that the two people concerned must be close to one another. The existence of telephones, of course, gives the lie to this supposed "precondition". It may be noted that the terminology used by Meehan's TALE-SPIN refers not to proximity, but to "D-LINK", which subsumes the various ways of establishing communication (including closeness). However, D-LINK is then definable simply - and only - in terms of enabling communication: and so stating that a precondition of an event is some state which enables that event; and this looks vacuous.

There are several other areas where the representation language is inadequate: the representation of distances and of durations has not been worked out. The fundamental difficulty is that precise measures exist for both; precise measures are seldom needed; yet sometimes only precise measures will suffice. How a system can coordinate its use of fuzzy and precise measures, and how it can determine which is appropriate for a given situation, are areas where further work must be done.

There is one problem with the inferencer itself which suggests that pronouns and WH-questions cannot be treated in identical fashion: when a speaker asks a question, it is permissible to question his presuppositions; when he uses a pronoun (in the context of a declarative sentence), there is some guarantee that a referent for that pronoun can be found. In AD-HAC, however, there is no provision for challenging presuppositions; and consequently, AD-HAC gives an answer to the final question in

JILL AND BILL WENT TO THE ZOO.
THEY GAVE THE MONKEYS SOME PEANUTS, WHICH THEY ATE.
THEY WENT TO THE RESTAURANT AND DRANK SOME TEA.
JILL TOOK BILL'S MONEY FROM HIM AND GAVE IT TO THE TRAMP WHO WAS TALKING TO THEM.

WHO TALKED TO THE MONKEYS?

(I am indebted to Bruce Croft for bringing this idiocy to light.)

11.5) Summary

AD-HAC tackles the problems of language processing from several fronts. The analysis of sentences is accomplished in an active fashion, using requests embedded in a nondeterministic preference-directed framework, using an ATN to locate and characterise low-level syntactic constituents. The inferential processing is then directed by a set of networks, keyed both on the conceptual primitives of the underlying representation language, and on tokens designating classes of real-world entities. The formulation of inference processes in this fashion brings many benefits, but especially provides a unified approach to the use of situation-specific, script-like, knowledge and mundane knowledge of the effects and preconditions of elementary actions. The production of inferences, and the subsequent compaction of the set of inferences drawn, provides a sound basis for the identification of pronoun referents and for the answering of questions about the texts.

Many problems have appeared in the course of this research. AD-HAC continues to provide a large-scale model in which research into these topics can be conducted.

Appendix A: the roles associated with CD primitives

This appendix shows the roles associated with individual ACT primitives, and also those associated with unusual states, that is, those which do not fit into the THING and VAL pattern.

The notation used here is as follows: the ACT or STATE primitive is underlined, and is followed by the roles it can take. Where a role is optional, it will be preceded by "OPT:"; where several roles may occur, but are mutually exclusive, they will be separated with "/".

ATRANS
    ACTOR
    OBJECT
    TO
    FROM
    TRUTH/ABILITY
    TIME
OPT: INST
OPT: MANNER

PTRANS
    ACTOR
    OBJECT
    FROM
    TO
    TRUTH/ABILITY
    TIME
OPT: INST
OPT: MANNER

MTRANS
    ACTOR
    MOBJECT
    FROM/FROMCP/FROMLTM
    TO/TOCP/TOLTM
    TRUTH/ABILITY
    TIME
OPT: INST
OPT: MANNER

PROPEL
    ACTOR
    OBJECT
    FROM
    TO
    TRUTH/ABILITY
    TIME
OPT: INST
OPT: MANNER

INGEST

    ACTOR
    OBJECT
    TRUTH/ABILITY
    TIME
OPT: INST
OPT: MANNER

ATTEND

    ACTOR
    OBJECT
    TO
    TRUTH/ABILITY
    TIME
OPT: INST
OPT: MANNER

MBUILD

    ACTOR
    MOBJECT
    FROM1
    FROM2
    TRUTH/ABILITY
    TIME
OPT: INST
OPT: MANNER

SPEAK

    ACTOR
    SOUND
    TRUTH/ABILITY
    TIME
OPT: INST
OPT: MANNER

MFEEL

    ACTOR
    EMOTION
    TO
    TRUTH/ABILITY
    TIME
OPT: INST
OPT: MANNER

GRASP

    ACTOR
    OBJECT
    TRUTH/ABILITY
    TIME
OPT: INST
OPT: MANNER

<u>MOVE</u>

ACTOR
OBJECT
TRUTH/ABILITY
TIME
OPT: INST
OPT: MANNER

<u>DO</u>

ACTOR
OPT: OBJECT
TRUTH/ABILITY
TIME
OPT: INST
OPT: MANNER

<u>ACT?</u>

ACTOR
OPT: OBJECT
TRUTH/ABILITY
TIME
OPT: INST
OPT: MANNER

<u>EXPEL</u>

ACTOR
OBJECT
TRUTH/ABILITY
TIME
OPT: INST
OPT: MANNER

<u>LOC</u>

THING
VAL
OPT: SPATREL
TRUTH

<u>MLOC</u>

MOBJECT
IN/INLTM/INCP
OPT: CERTAINTY
TRUTH
TIME

<u>GOAL</u>

WANTER
GOALSTATE
TRUTH
TIME

<u>FAMILY</u>

THING
VAL
RELATION
TRUTH
TIME

<u>EXIST</u>

THING
TIME
TRUTH

<u>IDENTITY</u>

THING
VAL

<u>CAUSE</u>

ANTECEDENT
RESULT
TRUTH/ABILITY/UNKNOWN
OPT: TIME

<u>CONJUNCT</u>

FIRST
SECOND
TRUTH/ABILITY/UNKNOWN

All states other than those listed above take

THING
VAL
TIME
TRUTH

Appendix B: The analyser's vocabulary

69 Adjectives

ANCIENT	ANGRY	ANNOYED	ANXIOUS	APPREHENSIVE
BIG	BLOATED	BLUE	BRAINY	BROKEN
BROWN	CALM	CAREFREE	CLEVER	CROSS
DAMAGED	DEAD	DRUNK	ENORMOUS	FEARFUL
FOOLISH	FURIOUS	GREEN	GUILTY	HAPPY
HEALTHY	HEAVY	HUNGRY	IMPENITENT	IMPOLITE
KAPUT	LACKADAISICAL	LARGE	LIGHT	LITTLE
LIVID	LONG	MORONIC	OBDURATE	OFFHAND
OLD	PARALYTIC	PECKISH	PENITENT	POLITE
RAVENOUS	RED	REMORSEFUL	RUDE	RUINED
SAD	SHORT	SICK	SILLY	SMALL
SMART	SOBER	STUPID	THICK	THIRSTY
TINY	TIPSY	TRANQUIL	UNWELL	WELL
WHOLE	WORRIED	YELLOW	YOUNG	

7 Conjunctions

AFTER	AND	BECAUSE	BEFORE	UNTIL	WHEN	WHY

89 Common nouns

AIR	ALCOHOL	BANANA	BED	BILL
BISCUIT	BOOK	BOY	BROTHER	BUILDING
BUS	CAN	CAR	CAT	CHILD
COFFEE	COMPUTER	CROOK	CROWN	CUP
DALE	DAUGHTER	DESK	DOCTOR	DOG
DUCK	ELEPHANT	FATHER	FOOD	FOOT
GIRL	GRANDMOTHER	GRASS	HAIR	HAMMER
HAND	HANDLE	HEAD	HILL	HOME
HUSBAND	JOHN	KEY	LORRY	MAN
MILK	MILKMAN	MONEY	MONKEY	MOTHER
NOTICE	NUT	OFFICE	PAIL	PAPER
PARK	PEANUT	PIE	PLANE	PLUG
PROGRAM	RESTAURANT	RIVER	ROAD	ROOM
SHELF	SIGN	SISTER	SKIN	SMOKE
SON	STORY	TAXI	TEA	TELESCOPE
TERMINAL	TOOTH	TOP	TOY	TRAMP
TREE	TRUCK	VINEGAR	WATER	WELL
WIFE	WINDOW	WOMAN	ZOO	

27 Proper names

ARTHUR	BEN	BILL	BRAN	BRIAN
CAROLINE	DALE	DAVID	ELAINE	ERICA
FRED	GEORGE	JACK	JANE	JILL
JOHN	KAREN	KEVIN	MARIEL	MARTYN
MARY	PAUL	RICHARD	RYLLAN	SPUD
STEVE	SUSAN			

21 Prepositions

AGAINST	ALONG	AT	AWAY	BY
DOWN	FOR	FROM	IN	INTO
NEAR	OF	OFF	ON	ONTO
OUT	OVER	TO	UNDER	UP
WITH				

111 Verbs

ADMIT	ADVISE	APOLOGISE	ARRIVE	ASK
BE	BEAT	BECOME	BEGIN	BELONG
BLUSH	BREAK	BRING	CARESS	CATCH
CHASE	COME	COMPLAIN	CONVINCE	CRASH
CRY	DECIDE	DENY	DEPRESS	DIE
DISPLEASE	DISTIL	DO	DRINK	DROP
EAT	ENJOY	EXHALE	EXPECT	FALL
FAST	FEED	FEEL	FETCH	FLOAT
FORGET	FRIGHTEN	GET	GIVE	GO
HASSLE	HATE	HAVE	HEAR	HIT
HOLD	HURT	INJURE	KICK	KILL
KISS	KNOW	LAUGH	LEAVE	LIE
LISTEN	LOOK	LOVE	MAKE	MEET
MEMORISE	MEND	OWN	PERSUADE	PLACATE
POUR	PREVENT	PULL	PUSH	READ
REALISE	RECALL	RECEIVE	REFUSE	REMEMBER
REPAIR	RIDE	ROB	RUN	SAY
SEE	SEND	SING	SMOKE	SPIT
START	STEAL	STOP	STRIKE	SWERVE
TAKE	TALK	TELL	THANK	THINK
THREATEN	THROW	TICKLE	TRAVEL	TUMBLE
UNDERSTAND	WALK	WANT	WATCH	WEEP
YAWN				

8 Determiners

A AN SOME THAT THE THESE THIS THOSE

11 Quantifiers

ALL	ANY	EACH	EVERY	FEW
LOTS	MANY	NO	NONE	SEVERAL
SOME				

18 Pronouns

HE HER HIM HIS I IT ITS ME MY OUR SHE THEIR THEM THEY US WE YOU YOUR

7 relative pronouns and WH-pronouns

THAT WHERE WHICH WHO WHOM WHAT WHOSE

8 quantified pronouns

ANYBODY ANYTHING ANTWHERE ANYONE NOBODY NOTHING NOWHERE NO-ONE

Various modals, such as CAN, WILL, WON'T

Appendix C: Pronoun reference, definite reference and question answering

This appendix shows the pronoun-reference processes deciding on the answer to the question "What did she give him". The story was processed in normal fashion up until the final sentence was to be read; the function MAIN was left, some debugging switches thrown, and the program restarted. We pick up from that point:

```
*(MAIN I)

The story so far:
BILL AND JILL WENT TO THE ZOO.
THEY GAVE THE MONKEYS SOME PEANUTS,WHICH THEY ATE.
THEY WENT TO THE RESTAURANT AND DRANK SOME TEA.
JILL TOOK BILL'S MONEY FROM HIM AND GAVE IT TO THE TRAMP WHO WAS TALKING TO
THEM.
Pray continue:
WHAT DID SHE GIVE THE TRAMP?
[]
DUMMY-FEMALE1 = JILL, score= 0.99832566 (0.027421660) ===>>> 0.99841497
..(TRAMP2=TRAMP1 ( 0.99245914,0.60000000)===>>> 0.99810944
..(?DUMMY-THING4=MONEY3 ( 0.99303074,0.20796244)===>>> 0.99542490
..(TIMEPOINT70=TIMEPOINT133 ( 0.65464701,0.45000000)===>>> 0.85327867
..(TIMESPAN2=TIMESPAN78 ( 0.78554802,0.15)===>>> 0.83693064
..(TIMESPAN52=TIMESPAN79 ( 0.78554319,0.15)===>>> 0.83692686
..(TIMESPAN77=TIMESPAN44 ( 0.78506829,0.15)===>>> 0.83655533
..(TIMESPAN5=TIMESPAN79 ( 0.78506829,0.15)===>>> 0.83655533
..(TIMESPAN9=TIMESPAN79 ( 0.78506829,0.15)===>>> 0.83655533
..(TIMESPAN30=TIMESPAN79 ( 0.77018507,0.15)===>>> 0.82488764
..(TIMESPAN78=TIMESPAN2 ( 0.78554802,0.00047972798)===>>> 0.78573165
..(TIMESPAN79=TIMESPAN52 ( 0.78554319,0.00047490000)===>>> 0.78572498
..(TIMESPAN44=TIMESPAN77 ( 0.78506829,0.0)===>>> 0.78506829
..(TIMEPOINT133=TIMEPOINT70 ( 0.65464701,0.24355421)===>>> 0.77468403
..(TIMESPAN59=TIMESPAN81 ( 0.57188409,0.15)===>>> 0.66485139
..(TIMESPAN82=TIMESPAN73 ( 0.54585102,0.15)===>>> 0.64318828
..(TIMESPAN80=TIMESPAN45 ( 0.54553377,0.15)===>>> 0.64292331
..(TIMESPAN45=TIMESPAN80 ( 0.54553377,0.15)===>>> 0.64292331
..(TIMESPAN81=TIMESPAN59 ( 0.57188409,0.026033066)===>>> 0.58914601
..(TIMESPAN73=TIMESPAN81 ( 0.54585102,0.0)===>>> 0.54585102
..(TIMESPAN49=TIMEPOINT133 ( 0.41109279,0.15)===>>> 0.52850321
..(TIMESPAN47=TIMEPOINT133 ( 0.41109279,0.15)===>>> 0.52850321
..(TIMEPOINT88=TIMEPOINT133 ( -0.18150000,0.15)===>>> -0.030665142
..(TIMESPAN35=TIMESPAN79 ( -0.64350000,0.15)===>>> -0.45005814
..(TIMESPAN13=TIMESPAN79 ( -0.64350000,0.15)===>>> -0.45005814
Do you want to see why? *Y

        ; The effect of these debugging switches is to make the program
        ; dump out some information just before replacing a pronominal
        ; token with its referent. The information is shown above, but
        ; is rather meaningless at the moment. Basically, each specification
        ; is inspected, to see what possible identifications are suggested,
        ; and the certainty ratings of the associated propositions are
        ; taken into account. Let's see more...
GET-RESOLUTIONS for DUMMY-FEMALE1
        ; the program is deciding to identify DUMMY-FEMALE1 (= "SHE") with
        ; something
Potential resolution: TRAMP1
        ; TRAMP1 might be the answer ... (Tramps are not always male)
```

```
Estimated agreement:  0.97090400
        ; TRAMP1 scores 0.971 approx, because
Derived from: ((0.78506829         ; there is some agreement between
                 (EXIST34          ; this proposition
                   EXIST31         ; and this proposition. The specifications
                   ((DUMMY-FEMALE1 . TRAMP1)         ; have two links
                     (TIMESPAN78 . TIMESPAN44))))   ;
               (0.54585102         ;there is some agreement between
                 (POSS22           ; these two as well
                   POSS31
                   ((DUMMY-FEMALE1 . TRAMP1)
                     (?DUMMY-THING4 . MONEY3)
                     (TIMESPAN81 . TIMESPAN73))))
               (0.41109279         ; and between these two
                 (MLOC45
                   MLOC15
                   ((GOAL37 . CONCEPTS1)
                     (DUMMY-FEMALE1 . TRAMP1)
                     (TIMEPOINT133 . TIMESPAN47)))))
        ; the numbers associated with each pair of propositions take
        ; into account both the certainty ratings, and the number of other
        ; identifications that would be needed for the propositions to be
        ; identical. These numbers are then combined, using AGREE, to give
        ; an overall measure of the evidence for the identity of these
        ; two tokens.

Potential resolution: JILL
        ; Another possibility, but this one has far more evidence, all
        ; positive, and so the overall measure is higher.
Estimated agreement:  0.99832566
Derived from: ((0.78554802
                 (EXIST34
                   EXIST23
                   ((DUMMY-FEMALE1 . JILL)
                     (TIMESPAN78 . TIMESPAN2))))
               (0.57188409
                 (POSS22
                   POSS29
                   ((DUMMY-FEMALE1 . JILL)
                     (?DUMMY-THING4 . MONEY3)
                     (TIMESPAN81 . TIMESPAN59))))
               (0.54553377
                 (LOC46
                   LOC12
                   ((TRAMP2 . TRAMP1)
                     (DUMMY-FEMALE1 . JILL)
                     (TIMESPAN80 . TIMESPAN45))))
               (0.46642835
                 (ATRANS31
                   ATRANS25
                   ((TRAMP2 . TRAMP1)
                     (DUMMY-FEMALE1 . JILL)
                     (?DUMMY-THING4 . MONEY3)
                     (TIMEPOINT133 . TIMEPOINT70))))
               (0.41109279
                 (MLOC45
                   MLOC39
```

```
                        ((GOAL37 . CONCEPTS1)
                           (DUMMY-FEMALE1 . JILL)
                           (TIMEPOINT133 . TIMESPAN49))))
                  (0.19607823
                     (GOAL39
                        GOAL33
                        ((TIMESPAN85 . TIMESPAN76)
                           (TRAMP2 . TRAMP1)
                           (JOY11 . JOY9)
                           (DUMMY-FEMALE1 . JILL)
                           (TIMEPOINT133 . TIMEPOINT70))))
                  (0.079075995
                     (GOAL35
                        GOAL29
                        ((TIMESPAN83 . TIMESPAN74)
                           (?DUMMY-THING4 . MONEY3)
                           (DUMMY-FEMALE1 . JILL)
                           (POSS37 . POSS33)
                           (TIMEPOINT133 . TIMEPOINT70)))))

      ; so JILL is a better candidate than TRAMP1
Do you want a full dump? (Y/N)
*N    ; don't waste paper
 ==== suggesting JILL (absolute 0.99832566 , relative 0.027421660)
       ; the first number given is the overall measure, as described above.
       ; the second number is the difference between this selected
       ; candidate and its closest competitor.
       ;
       ; All this was done before, and the result of AGREEing these two
       ; numbers gave the JILL=SHE resolution a higher total score than
       ; and other identification.
       ;
       ; Now we go back, and actually replace all occurrences of
       ; DUMMY-FEMALE1 with JILL - and print out a resolution of the
       ; pronoun.


Pronoun resolved : IT WAS JILL WHO GAVE SOMETHING TO A TRAMP.
        ; now the process repeats. This time, these two spans are the
        ; most closely linked. Lets see why.
TIMESPAN78 = TIMESPAN2, score= 1.0 (0.15) ==>>> 1.0
..(TIMESPAN2=TIMESPAN78 ( 1.0,0.15)===>>> 1.0
..(TRAMP2=TRAMP1 ( 0.99736807,0.60000000)===>>> 0.99934136
..(?DUMMY-THING4=MONEY3 ( 0.99769238,0.21262408)===>>> 0.99850102
..(TIMEPOINT70=TIMEPOINT133 ( 0.77531331,0.45000000)===>>> 0.90838572
..(TIMEPOINT133=TIMEPOINT70 ( 0.77531331,0.18925307)===>>> 0.84114483
..(TIMESPAN52=TIMESPAN79 ( 0.78554319,0.15)===>>> 0.83692686
..(TIMESPAN77=TIMESPAN44 ( 0.78506829,0.15)===>>> 0.83655533
..(TIMESPAN44=TIMESPAN77 ( 0.78506829,0.15)===>>> 0.83655533
..(TIMESPAN5=TIMESPAN79 ( 0.78506829,0.15)===>>> 0.83655533
..(TIMESPAN9=TIMESPAN79 ( 0.78506829,0.15)===>>> 0.83655533
..(TIMESPAN81=TIMESPAN59 ( 0.77147954,0.15)===>>> 0.82590430
..(TIMESPAN59=TIMESPAN81 ( 0.77147954,0.15)===>>> 0.82590430
..(TIMESPAN30=TIMESPAN79 ( 0.77018507,0.15)===>>> 0.82488764
..(TIMESPAN80=TIMESPAN45 ( 0.74400002,0.15)===>>> 0.80424614
..(TIMESPAN45=TIMESPAN80 ( 0.74400002,0.15)===>>> 0.80424614
..(TIMESPAN79=TIMESPAN52 ( 0.78554319,0.00047490000)===>>> 0.78572498
```

```
..(TIMESPAN49=TIMEPOINT133 ( 0.58606023,0.15)===>>> 0.67658250
..(TIMESPAN82=TIMESPAN73 ( 0.54585102,0.15)===>>> 0.64318828
..(TIMESPAN73=TIMESPAN82 ( 0.54585102,0.15)===>>> 0.64318828
..(TIMEPOINT88=TIMEPOINT133 ( -0.21780000,0.15)===>>> -0.065655049
..(TIMESPAN35=TIMESPAN79 ( -0.64350000,0.15)===>>> -0.45005814
..(TIMESPAN13=TIMESPAN79 ( -0.64350000,0.15)===>>> -0.45005814
Do you want to see why? *Y

GET-RESOLUTIONS for TIMESPAN78
Potential resolution: TIMESPAN2
Estimated agreement:  1.0
Derived from: ((1.0 (EXIST34 EXIST23 ((TIMESPAN78 . TIMESPAN2)))))
        ; aha! this is one of the similar propositions which suggested that
        ; JILL = DUMMY-FEMALE1. Note that the specifications have changed -
        ; a side effect of resolving the pronoun.

Do you want a full dump? (Y/N)
*N
 ===== suggesting TIMESPAN2 (absolute 1.0 , relative 0.15)
       ; when there is no competition, or the only competitor is a lot
       ; worse than the selected pairing, a maximum "difference" of 0.15
       ; is used.
       ;
       ; No messages are produced for this identification; it would be hard
       ; to express anyway: "JILL EXISTED AT THE TIME SHE EXISTED"?
       ;
       ; Repeat the process...
TRAMP2 = TRAMP1, score= 0.99736807 (0.60000000) ===>>> 0.99934136
..(?DUMMY-THING4=MONEY3 ( 0.99769238,0.21262408)===>>> 0.99850102
..(TIMEPOINT70=TIMEPOINT133 ( 0.77531331,0.45000000)===>>> 0.90838572
..(TIMEPOINT133=TIMEPOINT70 ( 0.77531331,0.18925307)===>>> 0.84114483
..(TIMESPAN52=TIMESPAN79 ( 0.78554319,0.15)===>>> 0.83692686
..(TIMESPAN77=TIMESPAN44 ( 0.78506829,0.15)===>>> 0.83655533
..(TIMESPAN44=TIMESPAN77 ( 0.78506829,0.15)===>>> 0.83655533
..(TIMESPAN5=TIMESPAN79 ( 0.78506829,0.15)===>>> 0.83655533
..(TIMESPAN9=TIMESPAN79 ( 0.78506829,0.15)===>>> 0.83655533
..(TIMESPAN81=TIMESPAN59 ( 0.77147954,0.15)===>>> 0.82590430
..(TIMESPAN59=TIMESPAN81 ( 0.77147954,0.15)===>>> 0.82590430
..(TIMESPAN30=TIMESPAN79 ( 0.77018507,0.15)===>>> 0.82488764
..(TIMESPAN80=TIMESPAN45 ( 0.74400002,0.15)===>>> 0.80424614
..(TIMESPAN45=TIMESPAN80 ( 0.74400002,0.15)===>>> 0.80424614
..(TIMESPAN79=TIMESPAN52 ( 0.78554319,0.00047490000)===>>> 0.78572498
..(TIMESPAN49=TIMEPOINT133 ( 0.58606023,0.15)===>>> 0.67658250
..(TIMESPAN82=TIMESPAN73 ( 0.54585102,0.15)===>>> 0.64318828
..(TIMESPAN73=TIMESPAN82 ( 0.54585102,0.15)===>>> 0.64318828
..(TIMEPOINT88=TIMEPOINT133 ( -0.21780000,0.15)===>>> -0.065655049
..(TIMESPAN35=TIMESPAN79 ( -0.64350000,0.15)===>>> -0.45005814
..(TIMESPAN13=TIMESPAN79 ( -0.64350000,0.15)===>>> -0.45005814
Do you want to see why? *Y

GET-RESOLUTIONS for TRAMP2
       ; there is only one possible resolution, and there is very good
       ; evidence for it. So identify these two.
Potential resolution: TRAMP1
Estimated agreement:  0.99736807
Derived from: ((0.78506829
                  (EXIST7
```

```
                  EXIST31
                  ((TRAMP2 . TRAMP1)
                     (TIMESPAN77 . TIMESPAN44))))
              (0.74400002
                 (LOC46 LOC12 ((TRAMP2 . TRAMP1)
                               (TIMESPAN80 . TIMESPAN45))))
              (0.59518442
                 (ATRANS31
                    ATRANS25
                    ((TRAMP2 . TRAMP1)
                       (?DUMMY-THING4 . MONEY3)
                       (TIMEPOINT133 . TIMEPOINT70))))
              (0.54585102
                 (POSS16
                    POSS31
                    ((TRAMP2 . TRAMP1)
                       (?DUMMY-THING4 . MONEY3)
                       (TIMESPAN82 . TIMESPAN73)))))

Do you want a full dump? (Y/N)
*N
 ===== suggesting TRAMP1 (absolute 0.99736807 , relative 0.60000000)


      ; This is a definite reference which is being linked to a preexisting
      ; token! let's say what we're doing...
Definite reference: A TRAMP IS A TRAMP.
     ; oh dear, that isn't very helpful. Never mind, we know what it means.
     ;
     ;
     ; and so on, and on, until everything is compacted again.
TIMESPAN77 = TIMESPAN44, score= 0.99994234 (0.15) ===>>> 0.99995739
..(TIMESPAN44=TIMESPAN77 ( 0.99994234,0.15)===>>> 0.99995739
..(?DUMMY-THING4=MONEY3 ( 0.99947983,0.21441153)===>>> 0.99966347
..(TIMESPAN80=TIMESPAN45 ( 0.99844804,0.15)===>>> 0.99885267
..(TIMESPAN45=TIMESPAN80 ( 0.99844804,0.15)===>>> 0.99885267
..(TIMEPOINT70=TIMEPOINT133 ( 0.89231626,0.45000000)===>>> 0.95774222
..(TIMEPOINT133=TIMEPOINT70 ( 0.89231626,0.30625603)===>>> 0.94132862
..(TIMESPAN52=TIMESPAN79 ( 0.78554319,0.15)===>>> 0.83692686
..(TIMESPAN5=TIMESPAN79 ( 0.78506829,0.15)===>>> 0.83655533
..(TIMESPAN9=TIMESPAN79 ( 0.78506829,0.15)===>>> 0.83655533
..(TIMESPAN81=TIMESPAN59 ( 0.77147954,0.15)===>>> 0.82590430
..(TIMESPAN59=TIMESPAN81 ( 0.77147954,0.15)===>>> 0.82590430
..(TIMESPAN30=TIMESPAN79 ( 0.77018507,0.15)===>>> 0.82488764
..(TIMESPAN82=TIMESPAN73 ( 0.74344774,0.15)===>>> 0.80380921
..(TIMESPAN73=TIMESPAN82 ( 0.74344774,0.15)===>>> 0.80380921
..(TIMESPAN79=TIMESPAN52 ( 0.78554319,0.00047490000)===>>> 0.78572498
..(TIMESPAN49=TIMEPOINT133 ( 0.58606023,0.15)===>>> 0.67658250
..(TIMEPOINT88=TIMEPOINT133 ( -0.21780000,0.15)===>>> -0.065655049
..(TIMESPAN35=TIMESPAN79 ( -0.64350000,0.15)===>>> -0.45005814
..(TIMESPAN13=TIMESPAN79 ( -0.64350000,0.15)===>>> -0.45005814
Do you want to see why? *Y

GET-RESOLUTIONS for TIMESPAN77
Potential resolution: TIMESPAN44
Estimated agreement:  0.99994234
Derived from: ((0.99994234 (EXIST7 EXIST31 ((TIMESPAN77 . TIMESPAN44)))))
```

```
Do you want a full dump? (Y/N)
*N
 ===== suggesting TIMESPAN44 (absolute 0.99994234 , relative 0.15)
?DUMMY-THING4 = MONEY3, score= 0.99947983 (0.21441153) ===>>> 0.99966347
..(TIMESPAN80=TIMESPAN45 ( 0.99844804,0.15)===>>> 0.99885267
..(TIMESPAN45=TIMESPAN80 ( 0.99844804,0.15)===>>> 0.99885267
..(TIMEPOINT70=TIMEPOINT133 ( 0.89231626,0.45000000)===>>> 0.95774222
..(TIMEPOINT133=TIMEPOINT70 ( 0.89231626,0.30625603)===>>> 0.94132862
..(TIMESPAN52=TIMESPAN79 ( 0.78554319,0.15)===>>> 0.83692686
..(TIMESPAN5=TIMESPAN79 ( 0.78506829,0.15)===>>> 0.83655533
..(TIMESPAN9=TIMESPAN79 ( 0.78506829,0.15)===>>> 0.83655533
..(TIMESPAN81=TIMESPAN59 ( 0.77147954,0.15)===>>> 0.82590430
..(TIMESPAN59=TIMESPAN81 ( 0.77147954,0.15)===>>> 0.82590430
..(TIMESPAN30=TIMESPAN79 ( 0.77018507,0.15)===>>> 0.82488764
..(TIMESPAN82=TIMESPAN73 ( 0.74344774,0.15)===>>> 0.80380921
..(TIMESPAN73=TIMESPAN82 ( 0.74344774,0.15)===>>> 0.80380921
..(TIMESPAN79=TIMESPAN52 ( 0.78554319,0.00047490000)===>>> 0.78572498
..(TIMESPAN49=TIMEPOINT133 ( 0.58606023,0.15)===>>> 0.67658250
..(TIMEPOINT88=TIMEPOINT133 ( -0.21780000,0.15)===>>> -0.065655049
..(TIMESPAN35=TIMESPAN79 ( -0.64350000,0.15)===>>> -0.45005814
..(TIMESPAN13=TIMESPAN79 ( -0.64350000,0.15)===>>> -0.45005814
Do you want to see why? *Y

GET-RESOLUTIONS for ?DUMMY-THING4
Potential resolution: MONKEYS1
Estimated agreement:  0.78506829
Derived from: ((0.78506829
                  (EXIST21
                     EXIST11
                     ((?DUMMY-THING4 . MONKEYS1)
                        (TIMESPAN79 . TIMESPAN5)))))

Potential resolution: PEANUTS3
Estimated agreement:  0.78506829
Derived from: ((0.78506829
                  (EXIST21
                     EXIST9
                     ((?DUMMY-THING4 . PEANUTS3)
                        (TIMESPAN79 . TIMESPAN9)))))

Potential resolution: TEA1
Estimated agreement:  0.77018507
Derived from: ((0.77018507
                  (EXIST21
                     EXIST2
                     ((?DUMMY-THING4 . TEA1)
                        (TIMESPAN79 . TIMESPAN30)))))

Potential resolution: MONEY3
Estimated agreement:  0.99947983
Derived from: ((0.79516474
                  (ATRANS31
                     ATRANS25
                     ((?DUMMY-THING4 . MONEY3)
                        (TIMEPOINT133 . TIMEPOINT70))))
                 (0.78554319
```

```
                    (EXIST21
                       EXIST35
                       ((?DUMMY-THING4 . MONEY3)
                          (TIMESPAN79 . TIMESPAN52))))
                 (0.77147954
                    (POSS22
                       (POSS29 POSS18)
                       ((?DUMMY-THING4 . MONEY3)
                          (TIMESPAN81 . TIMESPAN59))))
                 (0.74344774
                    (POSS16
                       POSS31
                       ((?DUMMY-THING4 . MONEY3)
                          (TIMESPAN82 . TIMESPAN73)))))

Do you want a full dump? (Y/N)
*N
 ==== suggesting MONEY3 (absolute 0.99947983 , relative 0.21441153)
TIMESPAN79 = TIMESPAN52, score= 0.99999941 (0.15) ===>>> 0.99999957
..(TIMESPAN52=TIMESPAN79 ( 0.99999941,0.15)===>>> 0.99999957
..(TIMEPOINT70=TIMEPOINT133 ( 0.99997481,0.45000000)===>>> 0.99999044
..(TIMEPOINT133=TIMEPOINT70 ( 0.99997481,0.41391458)===>>> 0.99998956
..(TIMESPAN81=TIMESPAN59 ( 0.99961704,0.15)===>>> 0.99971693
..(TIMESPAN59=TIMESPAN81 ( 0.99961704,0.15)===>>> 0.99971693
..(TIMESPAN80=TIMESPAN45 ( 0.99844804,0.15)===>>> 0.99885267
..(TIMESPAN45=TIMESPAN80 ( 0.99844804,0.15)===>>> 0.99885267
..(TIMESPAN82=TIMESPAN73 ( 0.99337692,0.15)===>>> 0.99510045
..(TIMESPAN73=TIMESPAN82 ( 0.99337692,0.15)===>>> 0.99510045
..(TIMESPAN49=TIMEPOINT133 ( 0.58606023,0.15)===>>> 0.67658250
..(TIMEPOINT88=TIMEPOINT133 ( -0.21780000,0.15)===>>> -0.065655049
Do you want to see why? *N
TIMEPOINT70 = TIMEPOINT133, score= 0.99997481 (0.45000000) ===>>> 0.99999044
..(TIMEPOINT133=TIMEPOINT70 ( 0.99997481,0.41391458)===>>> 0.99998956
..(TIMESPAN81=TIMESPAN59 ( 0.99961704,0.15)===>>> 0.99971693
..(TIMESPAN59=TIMESPAN81 ( 0.99961704,0.15)===>>> 0.99971693
..(TIMESPAN80=TIMESPAN45 ( 0.99844804,0.15)===>>> 0.99885267
..(TIMESPAN45=TIMESPAN80 ( 0.99844804,0.15)===>>> 0.99885267
..(TIMESPAN82=TIMESPAN73 ( 0.99337692,0.15)===>>> 0.99510045
..(TIMESPAN73=TIMESPAN82 ( 0.99337692,0.15)===>>> 0.99510045
..(TIMESPAN49=TIMEPOINT133 ( 0.58606023,0.15)===>>> 0.67658250
..(TIMEPOINT88=TIMEPOINT133 ( -0.21780000,0.15)===>>> -0.065655049
Do you want to see why? *N
TIMESPAN81 = TIMESPAN59, score= 0.99961704 (0.15) ===>>> 0.99971693
..(TIMESPAN59=TIMESPAN81 ( 0.99961704,0.15)===>>> 0.99971693
..(TIMESPAN80=TIMESPAN45 ( 0.99844804,0.15)===>>> 0.99885267
..(TIMESPAN45=TIMESPAN80 ( 0.99844804,0.15)===>>> 0.99885267
..(TIMESPAN82=TIMESPAN73 ( 0.99337692,0.15)===>>> 0.99510045
..(TIMESPAN73=TIMESPAN82 ( 0.99337692,0.15)===>>> 0.99510045
..(TIMESPAN49=TIMEPOINT70 ( 0.58606023,0.15)===>>> 0.67658250
..(TIMEPOINT88=TIMESPAN81 ( -0.61050000,0.15)===>>> -0.42186748
Do you want to see why? *N
TIMESPAN80 = TIMESPAN45, score= 0.99844804 (0.15) ===>>> 0.99885267
..(TIMESPAN45=TIMESPAN80 ( 0.99844804,0.15)===>>> 0.99885267
..(TIMESPAN49=TIMEPOINT70 ( 0.58606023,0.15)===>>> 0.67658250
..(TIMEPOINT88=TIMESPAN59 ( -0.64670567,0.15)===>>> -0.45278307
Do you want to see why? *N
TIMESPAN49 = TIMEPOINT70, score= 0.58606023 (0.15) ===>>> 0.67658250
```

```
..(TIMEPOINT88=TIMESPAN59 ( -0.64670567,0.15)===>>> -0.45278307
Do you want to see why? *N
TIMEPOINT88 = TIMESPAN59, score= -0.64670567 (0.15) ===>>> -0.45278307
Do you want to see why? *Y

GET-RESOLUTIONS for TIMEPOINT88
Potential resolution: TIMESPAN59
Estimated agreement:  -0.64670567
Derived from: ((-0.64670567
                  (POSS18
                     POSS22
                     ((DUMMY-HUMAN1 . JILL)
                        (TIMEPOINT88 . TIMESPAN59)))))

Do you want a full dump? (Y/N)
*N
 ===== suggesting TIMESPAN59 (absolute -0.64670567 , relative 0.15)


Answering question (1): JILL GAVE SOME MONEY TO A TRAMP.
Pray continue:
(Returning to Lisp...
```

References

[Abe, 1980]
Abe,M.
"A Japanese sentence generator"
Computer Laboratory, University of Cambridge, 1980 (Unpublished Diploma dissertation)

[Bobrow and Raphael, 1974]
Bobrow,D.G. and Raphael,B.
"New programming languages for artificial intelligence research"
Computing Surveys 6, 1974, 153-174

[Bobrow and Winograd, 1977]
Bobrow,D.G. and Winograd,T.
"An overview of KRL, a knowledge representation language"
Journal of Cognitive Science 1, 1977, 3-46

[Bobrow et al, 1977]
Bobrow,D.G.,Winograd,T., and the KRL Research Group.
"Experience with KRL-0: one cycle of a knowledge representation language"
Proc. Fifth International Joint Conference on Artificial Intelligence, 1977, 213-222

[Boguraev, 1979]
Boguraev,B.K.
"Automatic resolution of linguistic ambiguities"
Computer Laboratory, University of Cambridge, 1979, TR 11 (PhD thesis)

[Carroll, 1979]
Carroll,J.M.
"Names and naming: an interdisciplinary review"
IBM Thomas J. Watson Research Centre, Yorktown Heights, Report RC7370, 1979

[Charniak, 1972]
Charniak,E.
"Towards a model of children's story comprehension"
Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1972, AI TR-266

[Chomsky, 1965]
Chomsky, N.
Aspects of the theory of syntax
Cambridge, Mass.: M.I.T, 1965

[Culicover, 1976]
Culicover,P.W.
"A constraint on coreferentiality"
Foundations of Language 14, 1976, 109-118

[Cullingford, 1978]
Cullingford,R.E.
"Script application: computer understanding of newspaper stories"
Department of Computer Science, Yale University, 1978, research report 116, (PhD thesis)

[DeJong, 1979]
DeJong,G.M.
"Skimming newspaper stories by computer"
Department of Computer Science, Yale University, 1979, Research report 104, (PhD thesis)

[Doran, 1965]
Doran,J.
"An approach to automatic problem-solving"
Machine Intelligence 1, 1965, 105-123

[Goldman, 1974]
Goldman,N.M.
"Computer generation of natural language from a deep conceptual base"
Computer Science Department, Stanford University, 1974, STAN-CS-74-461 and AIM-247

[Hayes1, 1977]
Hayes,Patrick J.
"In defence of logic"
Proc. Fifth International Joint Conference on Artificial Intelligence, 1977, 559-565

[Hayes2, 1977]
Hayes,Philip J.
"Some association-based techniques for lexical disambiguation by machine"
Department of Computer Science, Rochester University, 1977, TR 25

[Herskovits, 1973]
Herskovits,A.
"The generation of French from a semantic representation"
Computer Science Department, Stanford University, 1973, AIM-212

[Kay, 1964]
Kay,M.
"A parser for categorial grammars"
Santa Monica, Calif.: The Rand Corporation, 1964, RM-4293-PR

[Kay, 1973]
Kay,M.
"The MIND system"
Natural Language Processing (ed. Rustin), New York: Algorithmics Press

[Kuno and Oettinger, 1962]
Kuno,S. and Oettinger,A.G.
"A multiple-path syntactic analyser"
Information processing 62: Proceedings of IFIP Congress 62 (ed. Popplewell), Amsterdam: North-Holland, 1963

[Lehnert, 1978]
Lehnert,W.G.
The process of question answering
Hillsdale, N.J.: Lawrence Erlbaum Associates, 1978

[Marcus, 1980]
Marcus,M.
A theory of syntactic recognition for natural language
Cambridge,Mass.: M.I.T., 1980

[McDonald, 1977]
McDonald,D.
"Language generation: the linguistics component"
Proc. Fifth International Joint Conference on Artificial Intelligence, p142

[Meehan, 1976]
Meehan,J.
"The metanovel: writing stories by computer"
Department of Computer Science, Yale University, 1976, research report no 74

[Rieger, 1974]
Rieger,C.J.
"Conceptual memory: a theory and computer program for processing the meaning content of natural language utterances"
Computer Science Department, Stanford University, 1974, CS419 and AIM233, (PhD thesis)

[Rieger, 1975]
Rieger,C.J.
"Conceptual overlays: a mechanism for the interpretation of sentence meaning in context"
Proc. Fourth International Joint Conference on Artificial Intelligence, 143-150

[Riesbeck, 1974]
Riesbeck,C.
"Computational understanding: analysis of sentences and context"
Computer Science Department, Stanford University, 1974, STAN-CS-74-437 (PhD thesis)

[Schank, 1972]
Schank,R.C.
"Conceptual dependency: a theory of natural language understanding"
Cognitive Psychology 3, 1972, 552-631

[Schank, 1973]
Schank,R.C.
"Identification of conceptualisations underlying natural language"
Computer models of thought and language (eds. Schank and Colby), San Francisco: Freeman, 1973

[Schank, 1975]
Schank,R.C. (editor)
Conceptual information processing
Amsterdam:North-Holland, 1975

[Schank et al, 1975]
Schank,R.C. and the Yale A.I. Project
"SAM - a story understander"
Department of Computer Science, Yale University, 1975, Technical report 43

[Schank and Abelson, 1977]
Schank,R.C. and Abelson,R.P.
Scripts plans goals and understanding
Hillsdale,N.J.:Lawrence Erlbaum Associates, 1977

[Schmidt et al, 1978]
Schmidt,C.F., Sridharan,N.S. and Goodson,J.L.
"The plan recognition problem: an intersection of psychology and artificial intelligence"
Artificial Intelligence 11, 45-83

[Small, 1980]
Small,S.
"Word expert parsing: a theory of distributed word-based natural language understanding"
Department of Computer Science, University of Maryland, 1980, TR-954 (PhD thesis)

[Sussman and Winograd, 1971]
Sussman,G.J. and Winograd,T.
Micro-planner reference manual.
MIT, A.I. Laboratory, 1971, AI memo 203.

[Wilensky, 1977]
Wilensky,R.
"PAM - a program that infers intentions"
Proc. Fifth International Joint Conference on Artificial Intelligence, 1977, p15

[Wilks, 1973]
Wilks,Y.A.
"An artificial intelligence approach to machine translation"
Computer models of thought and language (eds. Schank and Colby), San Francisco: Freeman, 1973

[Wilks, 1975a]
Wilks,Y.A.
"A preferential, pattern-seeking, semantics for natural language inference"
Artificial Intelligence 6, 1975, 53-74

[Wilks, 1975b]
Wilks,Y.A.
"An intelligent analyzer and understander of English"
Communications of the ACM vol. 18 no. 5, 1975, 264-274

[Wilks, 1977]
Wilks,Y.A.
"Good and bad arguments about semantic primitives"
Department of Artificial Intelligence, University of Edinburgh, 1977, Research report 42

[Wilks, 1978]
Wilks,Y.A.
"Making preferences more active"
Artificial Intelligence 11, 1978, 197-223

[Wilson, 1975]
Wilson,D.S.
Presuppositions and non-truth-conditional semantics
New York: Academic Press, 1975

[Winograd, 1971]
Winograd,T.
Understanding natural language
Edinburgh University Press, 1971

[Woods, 1970]
Woods,W.A.
"Transition network grammars for natural language analysis"
Communications of the ACM vol. 10 no. 13, 591-606, 1970

[Woods, 1972]
Woods,W.A.
The lunar sciences natural language information system
Cambridge, Mass.: Bolt, Beranek and Newman, 1972

[Woods, 1977]
Woods,W.A.
"Shortfall and density scoring strategies for speech understanding control"
Proc. Fifth International Joint Conference on Artificial Intelligence,
1977, 18-26