American Journal of Computational Linguistics *Microfiche 49 : 2*

C L A M : A COMPUTER LANGUAGE MODEL

NICHOLAS J. S. DOBREE

International Language Centre
P. O. Box 155712
Beirut, Lebanon

Temporarily: Rose Cottage, Hindon, Wiltshire, England

This paper describes a program which translates English into French. It is difficult to delineate the subset which a program can deal with, so sample sentences are given.

The analyser is multiple path, single pass, akin to Woods' A.T.N. grammar. The syntax is dealt with by matching with templates; the semantics by the application of semantic restrictions to syntactically associated pairs of word meanings. To limit the number of paths, all available guns, syntactic and semantic, are brought to bear at every stage. The output is a list of disambiguated word meanings, formed into a tree structure but with semantic rather than syntactic relationships between them.

The French generation first makes appropriate tense changes, then finds the French word(s) and redistributes them if necessary. This may generate a French structure radically different from the English. Then the words are sequenced and put into the correct form.

The program consists of about 8,500 Fortran instructions and the processing averages about 15 seconds per word on a 360-40.

TABLE OF CONTENTS

ACKNOWLEDGEMENT

My thanks are due to the Managers of IBM Near East and IBM Lebanon and many members of the staff of IBM Lebanon, who have helped and encouraged me in every way. I am also indebted to Yorick Wilks, Margaret King and Walther Bischoff of the Fondazione dalle Molle, who have been very helpful with their advice and ideas.

INTRODUCTION

There are, I hope three reasons why CLAM will be of interest to computational linguists.

(1) It is a working model. This is not a "paper" containing "ideas". It is a description of a model which works. To be more specific, it is a description of a large program, written in FORTRAN, which runs on a 360-40. It accepts as input English text, carries out a syntactic and semantic analysis of it, stores the result, and translates it into good French.

(2) The subset of English which it is capable of analysing is, by present standards, extremely large. The vocabulary is about 1300 words, many of which have a variety of meanings. More important that the size of the vocabulary is, of course, the range of syntactic structures and, perhaps most significantly, the degree of complexity of sentences which can be dealt with. Increasing the length and complexity of sentences does not bring likelihood of combinatorial explosion. The amount of working store and computing time required to analyse a sentence is of the order of the number of words in a sentence, although of course it varies according to the number of meanings of the words and the types of syntactic structure involved.

(3) The program is continuously extensible. This extensibility applies first to the subset of language which can be analysed, secondly to the target languages into which translations can be generated, and thirdly to the uses to which the analysis of the text can be put. In other words, I believe that

the program embodies a sound method of syntactic and semantic analysis such as must be the basis of a computer language model.

Extension of the subset of language which can be analysed is a matter of addition and refinement. It can be stated with confidence that such extension can be achieved because nothing fundamentally different from what has already been achieved is involved. New syntactic structures, well formed or otherwise, can be incorporated, by addition partly to the files and partly to the program. Continual refinements can be made to the method of finding pronoun antecedents. This problem, which seems to be generally accepted as the most difficult single problem in analysis, will never be solved by one simple algorithm, and the fact that a particular program at any given stage of its development gives the wrong answer in a particular case, so far from invalidating the program, rather points the way to further refinements (cf. Wilks, June 1975). What is important is that the program should provide the tools which enable the refinement to be made, and CLAM does this.

Extension of the target languages involves applying to other languages the same method which is used to generate French. This can be done, and indeed part of the actual program used for French would be generally applicable. It will be interesting to attack a language outside the Indo-European group, and Arabic is the first one I have in mind, although how soon this can be done is a question of time and priorities.

The obvious use to which the analysis can be put other than

translation is a question-answer system, and work on this is at present in hand. A question-answer system must be based on an effective analyser, and it is believed that CLAM can provide this. However, I do not maintain that the analyser should be independent of the memory and inferencing part of the system. Obviously it should not be independent of the memory, since an analyser must create and use its own memory, and although it would be theoretically possible for the analyser to have one type of memory and the latter part of the program to have another, this would be a ludicrous arrangement. The same argument applies to inferencing, which again has to be performed by an analyser. Therefore it seems that a question-answer system should be more integrated than many A.I. researchers appear to allow. On this score, I support the view of Wilks vis-a-vis Charniak.

To create a question-answer system, and, indeed, to improve the translation program, the memory and the semantics of the present program have to be developed. I use the word "developed" advisedly because I believe that the existing memory and semantics form a sound basis upon which a more comprehensive system can be built.

RESULTS

Assessment of the Subset of Language which the Model can Analyse

It is normal practice when describing a language model to leave discussion of the achievements of the model until the end.

First comes the description of how the model does or would operate, then, if it is actually in operation, an account of what it can do. In this description I am reversing the procedure, because I would not like to think of a reader ploughing through details of how something is done if he subsequently comes to the conclusion that what was done was not worth doing anyway. Let him first see what can be done, and then decide whether it is worth the trouble of reading on to discover how it is done.

Having said this, I am immediately confronted by the problem adumbrated by Woods of how a reader can assess the range and scope of a particular model, and by implication, of how the programmer can honestly present it. There are two standard methods of presentation. One is by rather sweeping general statements such as "the program can cope with noun clauses, adjectival clauses, conjunction, questions" etc., according to what claims are being made. Such generalisations are inevitably suspect and rightly so, since no reader will believe that he could not find, for example, adjectival clauses which the program could not cope with. The alternative method of presentation is to give sample sentences which the program has coped with, and hope that the reader will make for himself the type of generalisation which the programmer has scrupulously avoided. If the first method is adopted, the programmer may justifiably be branded as a charlatan. If the second, he runs the risk of having his sentences dismissed as "a few examples".

The problem is real, and the solution far from obvious: how to define a subset of language. Supposing that we were concerned only with single sentences and not longer texts; and supposing that it were possible, which evidently it is not, to list all the sentences of the subset: then how can we find a definition which would include all the sentences which we have listed and exclude any which we have not listed? Two things are clear. The definition would be very long, and it would contain an agglomeration of embedded provisos. For example, the section on relative clauses might include something like this:

Relative clauses are admissible, provided that

1. they do not contain more than seven words;
2. they do not contain a passive verb
   unless (a) it is a verb of 'cooking'
   or (b) the clause is a 'subject' clause;
3. there is no word between the noun and the relative clause unless it is part of a supervening relative clause;
4. the noun is not part of a subsidiary clause unless the subsidiary clause is itself a relative clause
   provided that (a) the noun is not the object of the clause
   and (b) the noun is not a 'time' noun.

All of the above provisos are of a type which could well be applicable at any particular stage in the development of a program, although some may be more likely than others. The programmer's difficulty is that until he has tried an appropriate type of sentence, he probably will not realise the existence of a

particular limitation. The first indication of it is that the sentence doesn't work, and he then has to rack his brain to find out why not, and alter the program to eliminate the limitation, thereby enlarging the subset in that particular direction (hoping that he is not at the same time being so stupid as to reduce it in another). Therefore if a programmer asserts that his program can deal with e.g. relative clauses of all types, he is probably not being dishonest but merely ignorant about the limitations of his own program. Whether such limitations should rightly be described as bugs, which Woods implies, is dubious, because that is tantamount to execting a program which can deal with some relative clauses to be able to deal with all relative clauses, and asserting that insofar as it cannot, something has gone wrong. Rather might one think of a program in terms of a pool of water spreading slowly over an area and gradually covering more and more of that area. The fact that the water covered a particular part of the area would carry no implication of covering any other part, although there would be a reasonable expectation of its spreading to a contiguous area next. This analogy, though valuable in helping to destroy a misconception, is evidently incomplete in two respects. It is two-dimensional, whereas language is multi-dimensional; and the program would advance not continuously, like a pool of water, but by fits and starts, in discrete steps. Each of these points is worth further examination.

Lip service has long been paid to the multi-dimensional nature of language, and yet the importance of this aspect in

attempting to analyse language has rather slowly come to be recognised. How many features are there which have to be taken into account, and what are they? How many possible relationships can exist between which of them? And, a question raised with particular force by computer analysers, what combinations of features are relevant? As a simple example, consider two features, both as it happens syntactic although the argument applies to semantic as well as syntactic features: relative clauses, and the passive voice. If a program can analyse each of the features separately, does it follow that it can analyse them in combination? Suppose, for example, it can analyse both of these sentences:

1. The man who came to dinner stole the silver.
2. The man was hit by a bus.

Does it follow that it can analyse this?

The man who was hit by a bus stole the silver.

Alas, it does not. It may in fact be able to, but there is no logical rule from which it can be deduced that it must be able to. Is the absence of suc a rule merely a computational quirk, or does it correspond to some linguistic truth? In this case, but not necessarily in all such cases, I would say that it does so correspond. We may ask ourselves if it is possible to imagine a language in which relative clauses exist, and the passive voice exists, but in which the verb in a relative clause cannot be in the passive. Of course it is, and there may for all I know be such languages. It is this absence of a logical rule of combination which makes the task of defining the bounds of a

subset of language so appallingly difficult, especially when it is remembered that it is not merely combinations of two features, as in the above example, but combinations of many features which have to be taken into account. Multi-dimensionality is such a cardinal characteristic of language that analysers dealing with tiny subsets from which this characteristic has been removed should probably be treated with reserve. They may give valuable insights, but they may also be misleading. I do not of course refer to the memory or inferencing part of microworld models, which is usually their raison d'etre, but only to their interface with natural language input. The designers of such models are inclined to regard the input analysis as little more than a tedious chore, and would be unlikely to take exception to what I am saying since they themselves normally make no great claims for this part of their models. But others, commenting on the models, sometimes make exaggerated claims on behalf of the analysers, and these claims should be guarded against. Perhaps the relationship between a language and a tiny subset of it with a strictly limited number of features should be thought of as akin to the relationship between a wall and a stone. They are recognisably composed of the same substance, but one has essential characteristics which the other totally lacks.

Before leaving the subject of multi-dimensionality, I would like to touch briefly on the possibility, at some time in the future, of devising a standard method of determining the extent of a subset of a language. The following idea could be considered, probably to be rejected, but at least it could provide a starting-

point for discussion. A number of features, say n, could be decided upon, the number varying according to the degree of subtlety of delineation required. An n-dimensional array with n+1 columns in each dimension (for the n features + 1 blank) would then contain an element corresponding to every combination of these features. Some of these elements would be irrelevant, since they would represent impossible combinations. The valid elements could be filled or not, according to whether the subset contained the combination of features which they represented. For a programmer building up the subset which his program was capable of analysing, such an array could provide both a measure of achievement and a guide to what was missing.

To return now to the analogy of the puddle, the second respect in which it was incomplete was that a program, as it develops, does not advance continuously, like water spreading, but by fits and starts, in discrete steps. Suppose, for example, that a programmer is testing a particular feature like, say, relative clauses, as that is the feature we have previously discussed. Sentences containing relative clauses have been entered repeatedly, and each time they have been rejected or analysed incorrectly. Then at last comes the moment of triumph and relief when, for the first time, the program takes in such a sentence and analyses it correctly. At that moment, the program has not merely edged forward, but it has leapt. In terms of the array postulated in the last paragraph, not just one but several elements will in all probability have been covered in one step. It will not be known for certain which elements, until

more testing has been done and more sentences tried. But just as it would be ludicrous to suppose that because a program can analyse one relative clause it can analyse all, it would be equally ludicrous not to expect a program which can analyse one relative clause to be able to analyse at least some others. There is a section of program common to all relative clauses, which has to work before any can be analysed correctly, and once that section is working in combination with any features, the liklihood is that it will work in combination with at least some others.

What follows from this? Firstly, that no reliable method at present exists for the designer of a language model to delineate the subset of language which his model can analyse. Secondly, that desirable as it undoubtedly is, for the benefit of both the designers of models and those who seek to assess their scope, to devise such a method, it is going to be extremely difficult to do so. Thirdly, that in this unfortunate state of affairs a designer can but fall back on the established system of presenting a list of sentences which his program has analysed correctly, and leave it to the reader to make his own assessment of where the bounds of the subset analysable by the model lie. To discuss the sentences as simply a few examples would be unintelligent; equally unintelligent to see in them visions of universality. Where, between these two extremes, the reader's judgement falls should depend upon the variety of the sentences, and upon their complexity. If the program can deal with complexity in any area, it should be some indication of its power,

perhaps yet unrealised, to do so in other areas. It would be a sign of its versatility, of its ability to disentangle elaborate patterns and resolve them into their elementary components.

After this rather prolix introduction, I come eventually to my own "list of sentences" that have been successfully analysed by CLAM. They fall into two categories: those which have been translated into French, in which case the French translation is given; and those which have simply been analysed syntactically, and semantically and reduced to a base form. This is because during the last year I have not been working on the French generator but concentrating on certain aspects of the analyser; and so in order to save computer time, the French generation has been omitted. Thus the sentences without translation have been processed last.

The question arises of what exactly is meant by "analysed syntactically and semantically" and "reduced to base form". This will be more fully explained in the subsequent text. At this stage it is sufficient to say that a syntactic tree has been formed and semantic ambiguities resolved, and that semantic relations between words in the tree have been determined (e.g. a syntactic subject of a passive verb is recorded as the semantic object). Single word meanings are retained as basic units. There is no Schankian-type resolution into semantic primitives, except insofar as this is implicit in the classification system. This is the base form from which the French has been generated. It has not so far proved necessary to go any baser. Development, as will be explained later, is envisaged along the lines of extending

the network rather than breaking down the units.

SAMPLE SENTENCES

The following are samples of sentences which have been correctly analysed by the program. They are given, together with the French translations where these have been produced by the program, and with comments on points of interest in the sentences.

1. The shirt which you sold is dirty.

   La chemise que vous avez vendue est sale.

   Relative clause.

2. The man and woman doctors saw have eaten the bread.

   L'homme et la femme que les medecins ont vu ont mange le pain.

   Contact clause (relative clause with relative pronoun missing).

   Simple conjunctive phrase.

   No article in English but article required in French.

3. I want the king to read the book.

   Je veux que le roi lise le livre.

   Accusative and infinitive.

4. I thought she would eat.

   J'ai pense qu'elle mangerait.

   Object clause with "that" missing.

5. He hurt some donkeys last month.

   Il a fait mal a des anes le mois dernier.

   Multiple-word verb in French.

6. He went to see the house.

Il est alle voir la maison.

He lived to eat.

Il a vecu pour manger.

Different types of infinitives.

7. The watch will work when the mechanic finishes working.

La montre fonctionner quand le mecanicien finira de travailler.

Time clause: present tense in English becomes future in French.

"De" after "finir" followed by infinitive instead of gerund.

Different meanings of "work".

8. When did you open the door?

Quand es -ce que vous avez ouvert la porte?

Question.

9. Drink the milk faster.

Buvez plus rapidement le lait.

Command.

10. The men got up.

Les hommes se sont leves.

Two-word verb.

Reflexive.

Verb takes "etre".

11. The clever queen's uncle disagreed.

L'oncle de la reine intelligente n'a pas ete d'accord.

Possessive.

Position of adjective.

12. Peel the potatoes for your mother.

Epluchez les pommes de terre pour votre mere.

Multiple-word noun.

13. Teachers write plays in March in some countries.

Les instituteurs ecrivent des pieces en Mars dans des campagnes.

Semantic resolution of "in".

'Des campagnes' should be 'certains pays

14. He stood up to put the fire off.

Il s'est leve pour eteindre le chauffage.

Two-word verbs.

15. That waiter, fat and stupid, was breaking the plates.

Ce serveur gros et stupide cassait les assiettes.

Appositional adjectives between commas.

Continuous tense.

16. The man who drank the wine does not laugh.

L'homme qui a bu le vin ne rit pas.

Negative.

17. You frightened the man whose pen you stole.

Vous avez effraye l'homme dont vous avez vole la plume.

"Whose"-- difficult construction.

18. The woman who you swam with is happy.

La femme avec qui vois avez nage est contente.

Floating preposition at end of relative clause.

*

19. The woman looks depressed and bored.

La femme a l'air ennuye and deprime.

Semantic resolution of "looks"

'Ennuye' and 'deprime' should be feminine. My ignorance.

20. The Queens should have arrived.

Les reines auraient du arriver.

"Should have" -- difficult construction.

21 I had to learn to shout.

J'ai du apprendre a crier.

Semantic resolution of "had".

22. Your brother, you and I found and your father bought her horses.

Votre frere, vous et moi avons trouve et votre pere a achete ses chevaux.

Mixed conjunction.

23. If you had come you would have met him.

Vous l'auriez recontre si vous etiez venus.

Conditional clause.

Compound tenses.

Pronoun object.

Concord of past participle after "etre".

24. Picking flowers is wrong.

Cueillir des fleurs est mauvais.

Gerund subject.

25. The king is as large as a cow.

Le roi est aussi grand qu'une vache.

"As" comparative.

26. I have never behaved rudely since you allowed me to stay.

27. He prefers painting pictures to working.

28. As many as six aeroplanes took off.

29. Men bought the book and clock. They mended it. It often did work.

Pronoun resolution.

30. How good a game is tennis.

31. I know which house the man was living in.

32. I know how easily embarrassed you are.

33. How clean a brush did you sweep the room with.

34. Men can understand which book is best.

DESCRIPTION OF METHOD

Before going into some detail about the method used to achieve these results, I would like to say something about the danger of over-sophistication on the part of the reader. There is a natural tendency for researchers, on reading something new, to look for points of broad similarity with something, anything, that they have read before; and, having found it, to sit back with relief and feel absolved from reading any further. In a field in which vast amounts are being written, it is a proper self-defence on the part of the reader, but in A.1. in particular, it has its special dangers.

When one passes from the realm of pure ideas to the hard practicalities of writing a computer program, a subtle change of emphasis occurs. The ideas, all embracing they may have

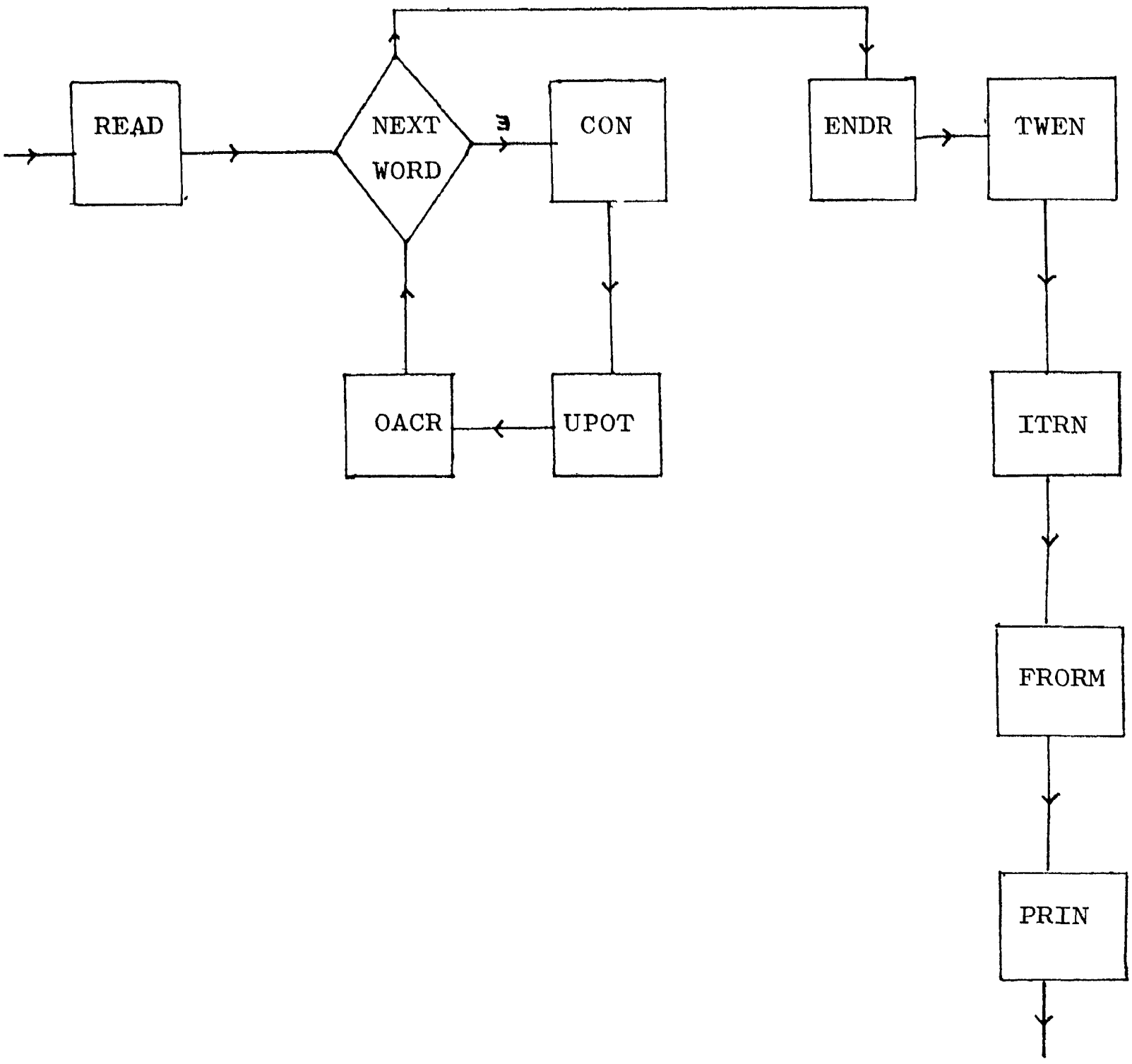
seemed at their inception, recede into the background, and what become vital are the details, the tiny mosaic pieces which determine whether the program succeeds. To judge a computer program by a crude classification of its method is like judging a picture by saying that it is impressionistic. Certainly it is impressionistic, but is it any good?

In case the reader is not convinced by this argument, let me say immediately that this is a multiple-path, single-pass, left-to-right, word-by-word analyser, akin to the multiple-path analyser of Oettinger and the augmented transition networ grammar of Woods. In order to tackle the semantics, and indeed also the syntactics, the meanings of words have been coded according to a hierarchical taxonomy That they are coded has been largely dictated by the demands of FORTRAN, in which the program is written, although some system more overtly like a networ could have been used. That the classification should be essentially hierarchical, with certain necessary refinements, has always seemed obvious.

Some details of how the program works now follow. I will start with the syntactic and semantic analysis, and come later to the generation of the French.

FLOWCHART

find codes syntactic and semantic analysis French



REDUCTION OF THE PASSAGE TO BASE FORM

A glance at the flowchart on the preceding page shows that there are two main parts of the program: first, the single subroutine READ, and second, a group of subroutines comprising the syntactic and semantic analysis.

READ This subroutine first transfers the base form of the preceding sentence to semi-permanent store. Then it reads the next sentence. It looks up each word in the dictionary file (VOCAB). If it cannot find it at first, it tests for certain endings such as -s, -ed, and -ing, subtracts them and tries again. When it finds the word it stores all the possible codes which are associated with the word in VOCAB. It also assembles compounds such as 'in front of', 'in order to', or infinitives, for which there is no single code. Proverbs or cliches can be similarly treated.

Coding Every possible meaning of a word has a code number containing a maximum of twelve digits. These code numbers are stored with the word in VOCAB and extracted in READ. The coding is based on a straightforward classification. For example the code of 'bull' is $\underset{\text{animal}}{6}\ \underset{\text{farm}}{2}\ \underset{\text{cow}}{1}\ \underset{\text{noun}}{2}\ \underset{\text{concrete}}{1}\ \underset{\text{creature}}{1}\ \underset{\text{male}}{1}$ Such classification is essential to reduce the number of syntactic and semantic patterns which have to be stored. It may be noted in passing that the system of coding contains the elements of both syntactic and semantic classification. The distinction between the two is at times tenuous. Further explanation of the coding is given in the appendix.

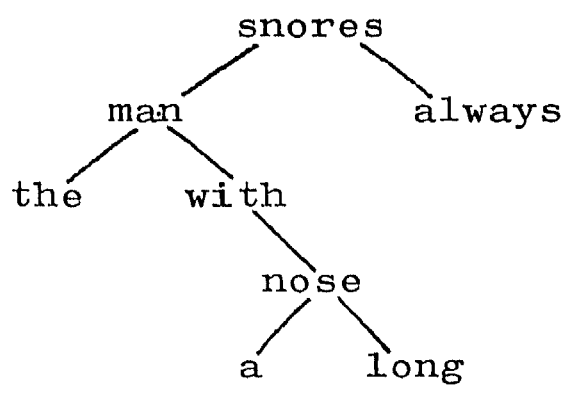
Syntactic and Semantic Analysis This is the most complicated part of the model, and comprises several sub-routines. For ease of explanation, many of them are here treated as parts of the larger routines CON, UPDT and OACR. As the flowchart shows, these three routines operate in turn on each word of a sentence, and when the end of the sentence is reached, a fourth routine, ENDR, is called on to operate on the complete sentence .

Before giving some account of the functions of these routines, it is necessary to explain the term EP, and to describe JEP and JSP, the two principle files referred to in this part of the program.

EP (English pattern). Take the sentence, The man with a long nose always snores. The program breaks this down into four EPs, as follows:

	EP 1	EP 2	EP 3	EP 4
lead word	snores	man	with	nose
subsidiary word	man	the	nose	a
subsidiary word	always	with		long

An EP contains one lead word plus a number of subsidiaries, and is classified according to the nature of the lead word. Thus EP 1 is a verb EP, EPs 2 and 4 are noun EPs, and EP 3 is a preposition EP. Man, the lead word of EP2, is a subsidiary of EP 1, so EP 2 is dependent on EP 1. Similarly, EP 3 is dependent on EP 2, and EP 4 on EP 3. Splitting a sentence into EPs is simply forming it into a tree structure.



JEP This file contains the templates for all the different types of EP. For example the template for a noun EP contains various types of adjective in appropriate sequence. These are followed by the lead noun. This is then followed by adjectives, appositional nouns, prepositions and relative pronouns. In an EP certain positions, such as the lead, are necessary, while others are optional. In a noun EP, the only necessary position is the lead noun. In a preposition EP, besides the lead preposition, the following noun is necessary. By far the most complicated EP is of course the verb EP. In this EP, later positions can be either closed, or opened, or made necessary, by a particular class of word in a particular position. For example, a pre-verb subject closes a post-verb subject. A question verb makes a post-verb subject necessary. One class of verbs opens a subsequent gerund position and closes a subsequent infinitive. Once an EP has been started, the program tests to see if the next word could occupy an open position on the template as far as the next necessary position (cf. below). Note that these templates are of syntactic patterns and bear no relation to Wilks' semantic templates.

JSP This file contains all the semantic patterns (SPs). An example of an SP is 12119 21 21102. This means that all verbs whose codes start with the digits 12119 can have as subjects any nouns whose codes start with the digits 21102. More specifically, it means that human beings read or write. In this case the verb would be the lead of a verb EP, and the noun would be a subsidiary word in the subject position of the EP. The middle group of digits in the SP specify the relationship between subsidiary and the lead. In this case, 21 specifies subject of verb. Similarly, SPs govern the relationships between the lead verb and all other subsidiary positions in the verb EP, and between lead and subsidiaries of all the other EPs. For example, 621 2 226 means that time prepositions, whose code words start with the digits 621, can have as objects any time nouns whose codes start with digits 226. When deciding whether a word is acceptable in a subsidiary position of a particular EP, a semantic match is made between that word and the lead word: JSP is searched to see if an SP exists permitting that word to be associated in that subsidiary position with that lead word (cf. below).

Processing the Sentence The sentence is processed in a single pass word by word from left to right. After each word, a number of possible continuation paths are open. The next word is tested along each of these paths, and if no place can be found for it that path is closed. If no places can be found, the path is reproduced n-1 times and the word added to each path Each path may then have one or more continuations.

Let us now return to the sentence, The man with a long nose always snores. The program goes through the sentence word by word, starting from the first. At the beginning, a verb EP is "open". That is to say, the program looks for all positions which could start a verb EP which the first word satisfies. In this case, 'the' cannot be part of a verb EP, but only of a noun EP, so the program will start a noun EP which is dependent on the subject position of a verb EP. The next word must continue the noun EP. Therefore on going to the next word, only EP2 is "open". 'Man' is then read, and EP2 and also EP1 are updated. At this point there are two alternative continuations. Either EP2 could be continued, as in fact happens, or EP2 could be "closed" and EP1 continued. Therefore on going to the next word, EPs 1 and 2 are both open. So the process is continued through the sentence.

As shown in the flowchart, there are three subroutines which operate on each word--CON, UPDT and OACR.

CON takes each EP which is open, and tests each sense of the word against each possible continuation of the EP. If the word could satisfy a position, it then looks to see whether a form match is necessary. In general, in English, a form test is only necessary between subject and verb, when the number and person must agree. If this hurdle is overcome, CON then proceeds to a semantic match. In general, the lead word of an EP must be matched semantically with every subsidiary word of that EP. For example a subject must be matched with a verb. So a check is performed, to see if that particular noun taken in that

particular sense could be the subject of that particular verb taken in that particular sense. Having found all the possible solutions, CON then gives way to UPDT.

UPDT updates each EP according to the solutions found in CON. It reproduces EPs as necessary where more than one solution has been found, and discards EPs which have become defunct because no solution has been found. It also determines which later positions of an EP either can or must be filled as a result of the current word becoming a part of the EP. It then hands over to OACR.

OACR (Open and Close Routine) determines which EPs must be kept "open" for the next word. It also performs some juggling with EPs in certain rather tricky cases such as relative clauses. It then returns control to the root program for the next word.

When all the words of the sentence have been processed, ENDR is entered. This examines all existing solutions. It discards any that are incomplete, and performs some housekeeping on those which are complete in order to separate them. In future, it will make a choice between alternative solutions, although this part of the program has not yet been written.

After ENDR, the sentence has been reduced to one (or more) sets of connected EPs. Within an EP, for each subsidiary word the relationship to the lead word (eg. verb/object, verb/time-noun, noun/article, etc.) is specified, as are the code(s) remaining as a result of the semantic matches which that word has undergone during the analysis.

After this brief description of the functions of the various subroutines, a more detailed explanation of the semantic match follows. We then show how the program deals with some of the more complex problems which it encounters.

Semantic Matching In order to illustrate the method, a simplified example is given, using the word 'in'. Take the sentence, She walked in fields in May. Suppose after READ, the following codes are in store:

```
 1..312.....1
 2..1141.....2
 3..1141.....2
 4..521.....3
 5..621.....3
 6..6216.....3
 7..6212.....3
 8..631.....3
 9..6311.....3
10..21274....4
```

The last digits, 1 to 6, refer to the word number. 'In' is words 3 and 5 with code numbers 4-9, 11-16. Suppose the codes have the same meanings as shown ascribed to 11-16. Suppose further that "places" start with digits 2127, and that 'field' is 21274, also that "time periods" start with digits 223, and that "months" start with 2235.

```
11..521.....5   adverb
12..621.....5   place preposition object - place
13..6211.....5  place preposition object - city
14..6212.....5  place preposition object - country
15..631.....5   time preposition object - time period
16..6311.....5  time preposition object - month
17..22355....6
```

It may well be asked why the distinction has been made between the three place prepositions and between the two time prepositions. There could be two reasons: either that the

concept of the preposition changes (which is probably not true here), or that the tranalation is different in some target language. If it is only the second case, the distinction could have been left for the program which generates the target language to draw. However, it is more economical to deal with it during the semantic matching.

Now let us see how the disambiguation process works. This example is simplified because it does not show the semantic matching across prepositions, between 'walked' and 'fields', and between 'walked' and 'May'. Although sometimes necessary for complete disambiguation, it is not so in this example, and as it complicates the explanation, I will omit it here for the sake of simplicity.

After the second word, there is only one EP open.

			code range	
EP1	lead	walk	2-2	(The meaning of "code range" will appear presently.)
	subject	she	1-1	

The third word, 'in', has two syntactic classes, adverb or preposition. Both are acceptable at this point in the verb EP. So a semantic match is performed between each class of 'in' and the lead word 'walk'.

Suppose that one SP gives 114 5 52,

another gives 11 6 62,

and another gives 11 6 63.

All the codes of 'in' are accepted-- code 4 by the first SP, codes 5, 6 and 7 by the second SP, and codes 8 and 9 by the third

The EP has to be reproduced because there are two syntactic classes of 'in'. We therefore have the following:

			code range				code ranges	
EP1	lead	walk	2-2	EP2	lead	walk	2-2	2-2
	subject	she	1-1		subject	she	1-1	1-1
	adverb	in	4-4		preposition	in	5-7	8-9

in EP2 there are two code ranges, one for the place preposition and one for the time prepositions. EP3, a preposition EP attached to EP2, is now opened, and for the next word this preposition EP and EP1 are open, but EP2 is closed.

The next word, 'fields', is a place noun. It is not accepted in EP1, which is therefore discarded. It is accepted in EP3 as the object of the preposition, so a semantic match is performed between 'in' and 'fields'.

Suppose there is an SP, 62 2 2127. Codes 5-7 are then accepted by this SP, and EP3 then looks like this:

			code range
EP3	lead	in	5-7
	object	field	10-10

A reconciliation is now carried out between the codes of 'in' in EP3 and EP2. As a result, the second code range in EP2 is eliminated.

The next word, 'in' again, is now read, and the process is repeated. EP2 now looks like this:

EP2	lead	walk	2-2	2-2
	subject	she	1-1	1-1
	preposition	in	5-7	5-7
	preposition	in	12-14	15-16

This time, on 'May', the relevant SP is 6311 2 2235.

There would also be an SP like this: 63 2 223.

But the first SP gives a narrower code range (16-16 instead of 15-16), and so it is preferred. This time, on reconciliation, the first code range in EP2 is eliminated and the second is reduced. So at the end, the three EPs are thus:

EP2	lead	walk	2-2	EP3	lead	in	5-7	EP4 lead in 16-16
	subject	she	1-1		object	fields	10-10	object May 17-17
	preposition	in	5-7					
	preposition	in	16-16					

We are now left with a code range for the first 'in' containing three codes. In such cases, it is the first code of the range which is selected. So 'in' has been disambiguated to 621 in the first case, and to 6311 in the second.

Syntactic Complexities Of course, it is all very well for a program to be able to digest, She walked in fields in May. But can it also cope with this?

The farmers we were talking about grew, and the green-grocers, thieves and liars, sold those apples.

In other words, the program must be capable of being expanded to deal with the myriad complexities and exceptions of natural

language. However sound the principles underlying a program may be, such expansion involves a deal of intricate and detailed work. At every stage,, flexibility and rigidity have to be balanced. The program must be flexible enough to envisage possibilities, but rigid enough to exclude impossibilities and to latch onto the right solution when it appears. The programmer's task resembles a tailor's. Let out an inch or two more, take in a couple there. It would be satisfactory indeed if an algorithm could be found both concise and comprehensive which would encompass all the requirements, but language is such a barnacled growth that this seems on the face of it improbable. It would be surprising if excrescences in the program were not necessary to deal with excrescences in the language. In the development of this program, when the treatment of a new structure has been added, whenever possible the original framework has been adapted to incorporate it, thereby avoiding the necessity of adding large sections of program. This is only commonsensical. Nevertheless, the program has grown considerably with its capacity to handle larger areas of language.

Here is perhaps a suitable point to emphasise that, since this is a multiple-path analyser, at each point all the available information, syntactic and semantic, has been deployed to eliminate incorrect paths. This has been done not only to avoid unnecessary computation, but also because the storage limits have made it essential. There are only 25 EPs. Frequently during testing this store overflowed, but interestingly enough

it has always been possible to bring the demand on it back within bounds by finding some restriction which had been overlooked and which cut out one of the paths. It had originally been feared that 25 EPs would not be nearly enough. One of the satisfying discoveries of the program is that it is.

Of course the deployment of all available information is not the only approach. Most of the earlier program concentrated on the syntax and paid little heed to the semantics. Wilks, on the other hand, is relying primarily on the semantics and is taking from the syntax only what is absolutely necessary. It will be fascinating if his research is able to determine exactly how much of the syntax is unnecessary. There are obvious redundancies in the form of unnecessary safe-guards in language. No one who has struggled with German case endings is ignorant of this. In English, we have the concord between subject and verb in the third person of the present, patently unnecessary since it exists only in this one instance. There are many sentences in which the semantics alone are clearly sufficient. In the sentence, "The man ate the steak with a fork.", the words could appear in any sequence and the meaning would be decipherable, although it might take longer to decipher. The interesting question is what features of the syntax can be consistently ignored, without occasional sentences cropping up which can only be deciphered with the help of these features.

There now follows a description of the treatment of three notoriously awkward problems-- relative clauses, pronouns, and conjunction.

Relative Clauses Six cases are distinguished:

1. The man who met you.
2. The man who(m) you met.
3. The man who(m) you gave it to.
4. The man to whom you gave it.
5. The man you met.
6. The man you gave it to.

After the lead of a noun EP, a relative pronoun (94), a preposition (6), and a contact noun (2R) are all possible continuations.

'Who' has three relative pronoun codes, starting with,

941, subject of relative clause,

942, object of relative clause,

943, object of preposition in relative clause.

'Whom' obviously only has the last two.

EP1 man	EP2 man	EP3 man	EP4 ---
the	the	the	(man) (subject)
who 941	who 942	who 943	
EP5 ---	EP6---	EP7 ---	
(man) (object)	---	(man) (preposition object)	

When a relative pronoun is recognised, the noun EP, EP1, is reproduced to EPs 2 and 3, and the codes 941, 942, and 943 are added to separate noun EPs. Then in OACR, new EPs 4 to 6 are opened dependent upon the noun EPs. In the case of 941 and 942, the lead of the noun EP, 'man', is entered in the new EPs as subject and object respectively. They are marked so as to avoid translation, but they are necessary for semantic matching in the relative clause. In the case of 943, an additional new

preposition EP, EP7, is opened dependent upon the relative clause EP, and the lead of the noun EP, 'man', is entered as the object of this preposition EP. The relative clause EP is marked as waiting for a floating preposition, although when a preposition comes this EP is reproduced, and in one EP the preposition is taken as the floating preposition, while in the other EP it is taken as another preposition. This is necessary to allow for such clauses as, the man whom you gave the book in the end to.

In the cases of 941, 942 and 943, the only EP which is open for the next word is the relative clause EP. For 941 the next necessary word in the EP is the lead verb, while for 942 and 943 the next necessary word is the subject. In practice, one or more of these EPs is usually eliminated on the next word.

When a contact noun is recognised, it is marked in the noun EP as being in reality a relative pronoun. Then the procedure for 942 and 943 above is followed; but in addition, the contact noun is entered as the subject of the relative clause EP.

When a preposition is recognised, the noun EP is reproduced once, because the preposition might be in the noun EP, like dog in a manger, or it might be in a relative clause. For the relative clause path, a preposition EP and a relative clause EP are opened. Only the preposition EP is left open for the next word, which must be a relative pronoun.

For indirect questions,

I don't know which house he bought.

I don't know what he lived in. etc

the treatment is somewhat similar to that for relative clauses.

Pronouns For either a translation or a question-answering program, the noun which the pronoun replaces, called here the replacement noun, has to be identified. In a question-answering program, the reasons are obvious enough. In a translation program, it is necessary for semantic matching and also because in many target languages the gender of the pronoun varies with that of the replacement noun.

The replacement noun might be in the same sentence as the pronoun, or in a previous sentence. Therefore, in dealing with pronouns, the program must be able to refer to preceding sentences. So after ENDR, the essential information for the sentence just processed is extracted from the first chain of EPs and stored. At present, this is only done for one chain of EPs, i.e. one solution. This essential information consists of a tree, containing one code for each word and the relation of each word to the code to which it is attached. Reverting to, The man with a long nose always snores., the information is as follows.

snores.....1.....1175

tense.....2.....tense, mood, code...T 1

man.....3.....211021.....1 1

the.....4.....4032.....3

with.....5.....617.....3

nose.....6.....212.....5

a.....7.....4033.....6

long.....8.....4176.....6

always.....9.....536.....1

The last column points to the code to which the word is attached. The previous column contains any relationship information not implicit in the code itself or, in the case of a pronoun, a pointer to the code of the replacement noun. It is important to notice that the code itself usually does provide the relationship information. For example 61, the first two digits of 'with', specify with some precision the relationship of 'with' to 'man'.

With the preceding sentences available in this form, the processing of a pronoun works as follows. When the pronoun is first encountered for a semantic match, all the possible replacement nouns are found; that is to say, all those nouns which agree in number and person with the pronoun and which are either before the pronoun in the same sentence but not in the same clause, or in a preceding sentence. The program only goes back through the preceding sentences until a suitable noun has been found. If for example there were one or more suitable nouns in the second sentence before the current one,

it would not examine the third sentence before the current one. Consider the following sentences.

The man went into the shop where he had seen the raincoat. He bought a hat and took it away.

For 'he', the only possible replacement noun is 'man' because it is the only noun which agrees in person. For 'it', the program finds 'hat', 'raincoat', and 'shop' as possible replacement nouns. If there were a preceding sentence, it would not bother to search it. Semantic matches are then carried out between 'take' and each of the three nouns and all three nouns are accepted, so they are all entered into the EP after 'it'. But the code ranges for 'shop' are more restricted than for 'hat' and 'raincoat', because the physical-movement meaning of 'take' is excluded with 'shop' because 'shop' is immoveable. When 'away' is read and matched with 'take', all meanings of 'take' except the physical-movement meaning are eliminated. 'Shop' is now left dangling, so to speak, and is eliminated as a possible replacement noun. So when the end of the sentence is reached, there are two possible surviving replacement nouns, 'hat' and 'raincoat'. There is no semantic reason for preferring one of these to the other, because the number of digits matched in the semantic match with 'take' is the same in both cases. Therefore in ENDR a choice is made according to a formula of priorities and 'hat' is selected, as a more recent verb object.

This "formula of priorities", which is only applied if there is no semantic preference for one noun, is probably at

the moment a rather blunt instrument. It is concerned with two factors -- which noun occurred in a later clause, and which noun has the same function as the pronoun; subject, object, preposition object, or object of the same preposition. In the majority of cases it produces the correct answer, but it is possible to think up examples in which it doesn't. With experience of use, the formula will be refined.

A complication is added by the possibility that, when a subject, 'it' may be impersonal. This sense is treated essentially as one possible replacement noun.

There is still work to be done in developing the formula of priorities. CLAM extracts the information required to solve the pronoun problem. The question is, how to use it.

Conjunction No part of the program is more complex than that dealing with conjunction. The principles are clear, even simple, enough; but applying them has demanded a considerable amount of care. Consider the fragment,

He cleaned the carpets in the bedroom and.....

When 'and' is read, the EPs are as follows:

EP1	EP2	EP3	EP4
cleaned	carpets	in	bedroom
he	the	bedroom	the
carpets	in		

All four of these EPs are alive, which is to say that the next word might be a continuation of any of them. On recognising a conjunction, the program looks for possible continuations in all alive EPs, from the beginning of the EP up to the point which has been reached. It carries out the necessary semantic

matches, it opens a new "conjep" or conjunctive EP for each solution, and it enters dummy words in both the conjep and the EPs above it in the chain where necessary. To clarify this procedure, we will consider two possible continuations.

(a).....and I..... 'I' can only be the subject of a verb EP, so the conjep, EP5, must be joined to EP1. The program adds a K entry, and opens EP5 thus:

EP1	cleaned	EP5-----
	he	I
	carpets	EP5 is dependent on EP1 at the subject position
	K5	

(b) .....and curtains. 'Curtains' could be joined to EP2 as the lead, or EP1 as the object. The conjep is attached to the lower EP, EP2, but a dummy word is entered in EP1 and the semantic match is carried out between the dummy word, 'curtains', and the lead of the EP, 'cleaned'.

EP1	cleaned	EP2	carpets	EP5 curtains
	he		the	the[x]
	carpets		in	
	curtains[x]		K5	EP5 is dependent on EP2 at the lead position.

'The' is entered as a dummy word in EP5 because it comes before the point at which EP5 is dependent on EP2. A semantic match is carried out between 'the' and 'curtains'.

'Curtains' might also be the subject of a verb EP, so EP1 is reproduced and another conjep started, attached to the reproduced EP at the subject position, as for.....and I..... above. This path is unlikely to be correct, and will probably

soon be eliminated.

An attempt is also made to attach curtains' to EP4 in the lead position, but it fails because a dummy word curtains' is then put into EP3, and the semantic match between 'in' and 'curtains' is tried and fails.

Now let us see what the EPs look like at the end of a more complex conjunctive sentence:

I, you and Nellie saw, watched and greeted the men, women and tired children.

EP1		EP2		EP5		EP6		EP7	
	saw		I		watched		greeted		men
	I		K3		I[x]		I[x]		the
	you[x]		K4[x]		you[x]		you[x]		K8
	Nellie[x]	EP3	you		Nellie[x]		Nellie[x]		K9
	K5		K4		K6		men[x]	EP8	women
	K6	EP4	Nellie		men[x]		women[x]		the[x]
	men				women[x]		children[x]		K9
	women[x]				children[x]			EP9	children
	children[x]								the[x]
									tired

It will be seen that control passes from the conjeps 5 and 6 up to EP1 before 'men', so that 'men' is entered as a word in EP1. But it is also entered as a dummy word in EPs 5 and 6, and semantic matches are carried out with 'watched' and 'greeted'. Also 'women' and 'children', although only dummy words in EP1, are entered as dummy words in EPs 5 and 6 as well.

A conjep remains open, and the EP on which it depends

remains closed until the last necessary word up to the branch has been filled. If the sentence had read,

I you and Nellie saw, and 'he' watched and greeted..etc.

EP5 would have opened with 'he'. 'I', 'you' and 'Nellie' would not have been entered in it as dummy words. EP5 would have remained open, and EP1 closed until after the lead word 'watched'.

A comma is treated as a possible conjunction or as a possible bracket. Because of the dual role of a comma, the programming associated with it is rather awkward.

To sum up the treatment of conjunction, the possible continuations from a conjunction, particularly if there have been previous conjunctions in the sentence, can be numerous. But by the strict use of dummy entries and their associated semantic matches, false continuations are usually quickly nosed out and eliminated. Also, for the recording of the full meaning of a conjunctive sentence for the purpose of later interrogation, the dummy entry system is of course essential. And in the special case of comparative sentences, it is only by such a system that it can be clearly established exactly what is being compared.

Summary. I conclude this section with an assessment of what the analysis can and cannot achieve. The purpose of analysis might be described as follows: to select, from among all the possible meanings of each word in the passage, its correct meaning in the context, and to determine what semantic relationships exist between which words. CLAM can do this

with considerable efficiency within the confines of a single sentence. It is just beginning to enlarge its horizens to deal with longer texts.

To clarify this statement let us consider the aids which enable us to select one meaning of a word rather than another, and see which of them CLAM applies.

1. Syntactic class. Example: "The car will work when the mechanic finishes his work." Here the word 'work' is evidently a verb on the first occasion and a noun on the second. CLAM can usually deal easily enough with this type of ambiguity.

2. Rules for pronoun antecedents. This has already been discussed at some length.. The rules are both semantic and syntactic. When the rules are determined, CLAM will be in a position to apply them.

3. Semantic restrictions on syntactically associated pairs of words which exclude one meaning. Example: "He took off his grandmother." Here the two word verb 'take off' must mean 'mimic'. The personal subject and the existence of an object excludes the sense of a plane taking off. 'Grandmother' as object excludes the sense of taking off clothes. Such restrictions are the basis of CLAM's semantic match, and ambiguities of this sort are resolved as a matter of course.

4. Semantic restrictions on syntactically associated pairs of words which give preference to one meaning. Example: "I killed the man with a gun." Here, there is a syntactic as well as a semantic ambiguity. It is less straightforward than

the previous example because the ambiguous word is 'with', which might be an instrument preposition attached to the verb 'kill', or a possession preposition attached to the noun 'man'. The semantic relationships which determine the choice, however, only involve 'with' indi ectly. They are between 'kill' and 'gun' in one case, and between 'man' and 'gun' in the other. Normally the preference would be for the instrument interpretation because 'gun' is more strongly associated with 'kill' as an instrument than with 'man' as a possession. CLAM chooses the stronger association by taking the 'deeper' semantic match, or in other words the match involving the larger number of digits. It does this correctly, but as we shall see in a moment, it is not always correct to do so.

5. Remoter contextual environment. Sometimes the factors enabling a choice to be made are more remote from the word in question than in the examples given above. In order to find these factors, a longer journey has to be made into the environment of the word.

Examples: (i) "The mayor hit the alderman so hard that he fell down." The normal rules for selection of pronoun antecedents would prefer 'mayor' as the antecedent of 'he' because it is the subject, but in the environment of hitting, it is much more likely to be the person hit who falls down rather than the hitter, so 'alderman' must be preferred.

(ii) "Two men came in. One had a gun and the other had a knife. I killed the man with a gun." Here 'with' is obviously not an instrument preposition attached to 'kill',

but a possession preposition attached to 'man'. This is so because the definite article 'the' attached to 'man' implies that 'man' has already been defined. But in fact two men have already been defined, and more information is needed to determine which of them is referred to. The only possible additional information which could satisfy this requirement is 'with a gun', which does suffice to distinguish one of the previously determined men. Therefore this phrase must be attached to 'man'.

At present, CLAM could not resolve either of these ambiguities. In order to do so it would need, in the first case, more information about the environment of 'hit' than is contained in the semantic restrictions now at its disposal, and in the second case, both a better memory and a routine for dealing with definition of nouns. Work is in progress on these vital additions. They will involve adding to the type and range of the semantic relationships between pairs of words referred to in the definition of the purpose of analysis given at the beginning of this summary. At present, CLAM only holds semantic relationships between words which are syntactically related. This is not enough. Adding to the types of relationships held, and extending them to pairs of words which are syntactically remote, will greatly increase the scope of the model.

GENERATION OF THE FRENCH TRANSLATION

As shown in the flowchart, the sentence is operated on sequentially by four subroutines--TWEN, ITRN, FRORM and PRIN. Briefly the function of each of these subroutines is as follows.

TWEN examines all the verbs. It welds them (joins auxiliaries to main verbs), and determines their tense in French. This is not of course necessarily the same as in English. Other features of the sentence often have to be examined. Thus, "When he arrives we will meet him", becomes in French, "When he will arrive we will meet him". And "I have been here for five years" becomes "I am here since five years." Gerunds, infinitives and participles are also dealt with by TWEN. It may well be asked why the weld part of this routine is thus left until the French generation. Should it not be done during the reduction of the English sentence to base form? The answer is that logically it should, and it will sooner or later be transferred, probably to ENDR. But at present it doesn't matter. The part of the program described in the section on pronouns which stores the base form of the last sentences is in fact performed after the French translation has been generated, and therefore, after the verbs have been welded.

ITRN takes each word in the sentence in turn. It finds the code number in FRILE, the French dictionary file, and extracts the French word(s). Sometimes of course there is

more than one. Sometimes there is zero because the English word does not have to be translated. Any particular French word may not have the same function in the sentence as the English word. In such cases, the French word entry in FRILE is followed by a code which specifies the word's function in relation to the English word being translated. For example, if 212237 is the code for 'potato', the FRILE entry will be 212237 POMME F DE 6 TERRE 6x2. The F after POMME shows that it is feminine. The 6 after DE shows that its function is as a preposition in the EP of which POMME is the lead. The 6x2 after TERRE shows that it is the object in the EP of which DE is the lead.

Sometimes it is necessary to go up the tree. For example Y1x5 means an adverb (5) in the verb EP (1) of which the English word is a subsidiary (Y). It is thus possible to generate a French sentence of a radically different shape from English.

ITRN also finds a French sequence code for each word. This is a code which provides the ordering of words within an EP. All lead words have the code 200. A pre-noun adjective may have a code 140, and a post-noun adjective 350. So these codes do not determine what is the actual sequence of words in the sentence, but they do provide the basic information from which the sequence is derived in FRORM.

FRORM first derives the actual sequence of words in the sentence. It then takes each French word and puts it into the

correct form. Obviously the most arduous part of this task is finding the forms of the verbs. FRORM refers to tables which contain the verb endings for both irregular and regular verbs, and the irregular feminine and plural endings for nouns and adjectives.

PRIN prints the French translation, having made any necessary elisions. If there is more than one solution, it prints alternative translations of particular words on subsequent lines or, if appropriate, it will print complete alternative sentences.

CONCLUSION

Programming Details and Future Developments

Programmers may be interested in some details. The program runs on a 360-40 using 146K of core store. The program is written in FORTRAN IV, not an ideal choice but the best available in the circumstances. The reduction of the English to base form requires about 6,000 instructions, and the French generation about 2,500. At present all the files are kept in core store except for the two large dictionary files VOCAB and FRILE, which are accessed on disk. It will eventually be necessary to keep JSP also on disk.

At present the processing takes about 15 seconds per word on average, of which READ takes 40%, the semantic and syntactic analysis about 20%, and the French generation 40%. No serious attempt has yet been made to optimise the program and this time could certainly be reduced. But the reduction would

be offset by the eventual need to keep JSP on disk. So as a practical proposition for translating texts, it would be necessary for the processing time to be reduced by a factor of about 10. Presumably this will come sooner or later with improvement in hardware.

There are certain improvements which would have to be made to the program before it could be used, apart from the extension of the vocabulary. Most obvious:

(a) there are some syntactic structures such as inversion after negatives which the program does not at present recognise;

(b) a selection routine must be incorporated in ENDR to choose between alternative solutions if more than one emerges;

(c) if no solution emerges the program should try again, selectively suppressing semantic matching, allowing words to be used outside their normal sense;

(d) the sizes of some of the temporary stores would have to be increased.

No particular difficulty is anticipated with any of these developments, in that they involve no methodology fundamentally different from what has already been applied. It is primarily a matter of time and priorities. However with a fifth development, namely the extension of the memory as outlined at the end of the section on analysis, new ground must be covered, and work on this is at present in progress.

APPENDIX

Coding System : Principal Categories

Digit	1st	2nd	3rd
	1 verb	1 intransitive	
		2 noun object	
		3 clause predicate	1 noun + part part.
			2 noun + to + infin.
			3 noun + gerund
			5 noun clause
		4 verb sequel	1 infinitive
			2 to + infinitive
			3 gerund
		5 noun + clause predicate	1 noun + infinitive
			2 noun + to + infin.
			4 noun + prep. + gerund
			5 noun + noun clause
		6 complement sequel	
		7 be (pres. cont.)	
		8	1 be (passive)
			2 have (pres. perf.)
	2 noun	1 concrete	1 animate
			2 inanimate
		2 abstract	
	3 pronoun		
	4 adjective	0 qualify concrete or abstract noun	

Digit	1st	2nd	3rd
		1 qualify concrete noun	0 animate or inanimate
			1 animate
			2 inanimate
		2 qualify abstract noun	
		4 question	
		5 possessive	
	5adverb	1 time	
		2 place	
		3 purpose	
		4 question	
		5 manner	
		6 degree	
		7 probability	
		8 frequency	
	6 preposition	1 predetermined	1 by verb
			2 by noun
			3 by adjective
		2 post determined	1 time
			2 place
			3 purpose
			4 reason
			5 manner
			6 instrument

Digit	1st	2nd	3rd
			7 association
			8 past
			9 concession
			A subject matter
	7 conjunction	1 link	
		2 contrast	
		3 comparison	
	9 clause word	2 noun clause	
		4 relative clause	
		6 adverbial clause	1 time
			2 place
			3 purpose
			4 reason
			5 manner
			8 condition
			9 concession