18

The Syntactic Analysis of English by Machine

J. P. Thorne
Department of English Language
P. Bratley and
H. Dewar
Department of Computer Science
University of Edinburgh

1. INTRODUCTION

In this paper we describe a program which will assign deep and surface structure analyses to an infinite number of English sentences.[1] The design of this program differs in several respects from that of other automatic parsers presently in existence. All these differences are a consequence of the particular aim we have pursued in writing the program, which represents an attempt to construct a device that will not only assign a syntactic analysis to any English sentence–that is, a record of the syntactic structure that the native speaker perceives in any English sentence–but which also, to some extent, simulates the way in which he perceives this structure. This is not to say that the analyzer differs from others because we have based its design upon the findings of psycholinguistic experiments. For one thing very few experiments on the perception of syntactic structure have been carried out and for the most part the results have been fairly inconclusive. But it is the case that we have, as far as possible, treated the task of constructing an automatic parser as being itself a psycholinguistic experiment. That is to say, any proposal regarding the possible operation of the program has been judged (mainly as the result of introspection) according to whether or not it seemed to be consistent with human behaviour. And this has led to our incorporating certain features which are absent from other automatic parsing systems.

Among the most notable of these features is the program's ability to assign syntactic labels to an infinite number of words while operating with a finite dictionary. As far as we know, all other automatic parsers of English (or

[1] This work was supported by the Office for Scientific and Technical Information Grant No. ID/102/2/06 to Professor Angus McIntosh. H. Whitfield and D. J. Dakin have also been associated with the work at various times.

Russian, etc.) are constructed on the assumption that they will eventually incorporate a dictionary containing every word in the English (Russian, etc.) language and listing the parts of speech to which each word can be assigned. It seemed to us essential that a syntactic analyzer should be able to deal with any sentence in the language without having to have access to such a dictionary, not only because the compilation of such a dictionary is a quite impracticable task, but because it seemed to us extremely unlikely that a speaker of the language needs to internalize and employ such a dictionary in order to recognize the syntactic structure of sentences in his language, if only because people are obviously able to recognize the syntactic structure of sentences containing words that they have never heard before.

Another important characteristic of the program is that it only needs to make one pass through the sentence it is analyzing and that any element in the sentence is analyzed once and once only. Again our reason for ensuring that the program should not call for multiple passes to be made through the sentence under analysis is the result of our conviction that under ordinary circumstances human beings do not need more than one, although (as with getting the program to operate with only a finite dictionary) it hardly surprised us to discover that this also had a significant bearing on the efficiency of the program.

But undoubtedly the most important decision that resulted from our attempt to construct a model for the perception of syntactic structure was our decision that the program should assign both deep and surface structure analyses to sentences. Our use of the terms 'deep structure' and 'surface structure' can be briefly (though inadequately) explained by the following example. Consider the sentence *The girl I liked left.* Any English speaker, having heard this sentence, possesses the information contained in the statement that in this sentence *The girl I liked* is the subject and *left* is the predicate, and that *The* is a definite article, *girl* a noun, etc. This is information about the surface structure of the sentence. But any English speaker also knows that in this sentence *The girl* is not only the subject of *left* but is also the object of *liked,* even though, of course, the correct surface form of the sentence is *The girl I liked left* not *The girl I liked the girl left* or *The girl I liked her left.* Since we derive this information from the sentence without making any reference to the context–as here where it is used merely as an example–then it is clear that we derive this information not from the context (as is sometimes suggested) but from our perception of the structure of the sentence, even though, as we have seen, this part of the structure is not actually realized in the surface form of the sentence. This information forms part of the deep structure of the sentence. Nearly all the automatic parsers now in operation give information only about the surface structure of sentences.[1]

[1] For accounts of other programs which assign deep structure analyses see Kay (1967), Petrick (1965), and Zwicky *et al.* (1965). For an explication of the concepts of deep and surface structure see Chomsky (1965).

2. METHODS AND APPROACH

Four constraints were originally imposed on the program:

(i) the program must not rely on looking up every word of an input sentence in a dictionary;

(ii) it must process each sentence in a single left-to-right pass;

(iii) it must be constructed in such a way that at every stage of the analysis process, some 'memory' would be remembering the decisions made up to the current point in the sentence, and some predictive process be looking ahead to what could possibly arrive next;

(iv) (in fact a consequence of (ii)) it must analyze each part of the sentence once and once only.

To these constraints we subsequently added a fifth:

(v) it must undertake deep and surface structure analyses simultaneously.

2.1

It does not seem to us reasonable to suppose that a person hearing or reading a sentence in any sense 'looks up each word in an internalized dictionary' in order to assign form-class information to it before proceeding with the analysis process. Several reasons can be advanced against such a hypothesis, perhaps the most cogent being that people can analyze sentences containing words which they have never heard before and which, therefore, they certainly cannot have in any internalized dictionary. For instance, nobody should have any difficulty in deciding the syntactic structure of the sentence *He has gone to shoot a grison*, although most people will not have heard of a *grison* before and will not know what it is. Again, with *He is going to disple his mother-in-law* or *She will be furibund*, although particular words may be unknown the syntactic structures of the sentences are clear. Indeed, far from its being the case that from a knowledge of the form classes to which particular words belong one deduces the structure of a sentence, it seems much more likely that from the rest of the structure of a sentence one can derive the classification of words in it. In the three instances: *He ruled with an iron hand*, *Strike while the iron is hot*, and *I will iron your shirt tomorrow*, the knowledge that *iron* can be an adjective or a noun or a verb would clearly be of no help in determining the complete syntactic structure of the sentences. In fact, it is because one recognizes the structure of the sentences that one knows that *iron* is an adjective in the first, a noun in the second, and a verb in the third.

This is not true of all words, however. It seems that words such as prepositions, pronouns, and conjunctions, which have fixed syntactic functions, play an essential part in the recognition of sentence structure. Words with fixed syntactic functions we call *closed-class words*, and all others *open-class words*, for the reason that all the former classes have a finite and, in fact, determinate

number of members while the latter have, in principle, an infinite number of members and are indefinitely extendable. Accordingly the program is designed to require access to a dictionary containing only the closed-class words.

2.2

Very many sentences are syntactically ambiguous. Nearly all sentences contain elements which taken separately are ambiguous. It is our conviction that usually in listening to such sentences all possible interpretations of these elements are considered simultaneously (certain kinds of jokes providing a possible exception). For this reason the analyzer is designed to go through each sentence in a single left-to-right pass. If any part of an input sentence is syntactically ambiguous, then all the possible analyses are developed simultaneously. That is to say, it is not the case that it first tries one analysis and then backtracks to see if any others are possible. The progress of the analysis process is recorded on a continually growing data structure, and when the end of the sentence is encountered, each possible analysis is to be found represented as a path through this structure.

2.3

There is a good deal of evidence to suggest that the efficiency with which human beings recognize the syntactic structure of sentences is to some extent the result of their ability, having heard part of a sentence, to predict the structure of the remainder. If one hears a sentence which breaks off suddenly in the middle, one is not left feeling that the sentence is ungrammatical but rather that the end of the sentence is missing. This seems to suggest that some kind of predictive mechanism is at work and that at some stage an expected outcome did not in fact occur. It seems likely that having heard (say) the subject of a sentence, we are then predicting, in some sense, the occurrence of a verb to go with that subject. If we look again at two of the examples given above, *He ruled with an iron hand* and *I will iron your shirt tomorrow*, then it is quite obvious that the kinds of words that can follow *He ruled with an . . .* and *I will . . .* are different. Roughly speaking, in the first case we are predicting either a noun or an adjective, while in the second case we are predicting a verb. Accordingly the operation of the analyzer is a process of making and checking predictions about syntactic structure. The source of information for these predictions is a representation of a grammar.

2.4

If it is reasonable to assume that in recognizing the syntactic structure of a sentence one considers all possible interpretations of ambiguous items simultaneously, then it seems equally reasonable to assume the converse–that unambiguous items are analyzed only once, no matter how many different possible analyses of the whole sentence or parts of the sentence they may enter into. Thus if the first part of a sentence has been analyzed in two different

ways and it then turns out that both analyses lead to an identical set of predictions, the rest of the sentence should be analyzed once only. It should not be necessary to produce two distinct but identical analyses for the rest of the sentence. Take for example the sentence *When he has fixed dates he will ring us*. There is an obvious ambiguity here (in one interpretation *has* is taken as an auxiliary, in the other as a main verb). But the ambiguity is confined to the first clause, and both analyses lead to identical predictions being made at the start of the second. Similarly in the sentence *He rolled up the bright red carpet*, the phrase *the bright red carpet* is either the object of the phrasal verb *rolled up* or the object of the preposition *up*. But the analysis of *the bright red carpet* as a noun phrase is the same for both interpretations. In these cases the program operates in such a way that analysis paths leading to the same predictions are conflated.

2.5

In Spring 1966 a first simple model incorporating all the features described above was implemented on KDF9. It had been deliberately designed to analyze only the surface structure of input sentences because at that time our idea was that this surface-structure analysis should then be used as the input to a separate deep-structure analyzer. Despite this limitation, the model (Thorne *et al.*, 1966) was extremely useful, as it enabled us successfully to test for the first time the feasibility of using only a closed-class dictionary, and of using a predictive technique conflating identical predictions.

But undoubtedly its main usefulness lay in its demonstrating to us that this approach was essentially wrong. The trouble with a two-stage analyzer, that is, one comprising two components, a surface structure analyzer and a deep structure analyzer, the output of the first being the input to the second, is that in the case of many types of sentences the surface structure analyzer produces a large number of incorrect analyses which the deep structure analyzer has to discard. This is particularly noticeable in the case of sentences containing embedded clauses or conjunctions like *and* and *but* (where the crucial factor is the surface structure analyzer's inability to take account of deletions in the deep structure). Any *ad hoc* attempts to reduce the number of analyses turned out to have the undesirable consequence that in many cases the number was reduced to zero. It became clear to us that the large number of wrong analyses produced by the surface analyzer was a direct result of the fact that it had to work independently, without access to any deep structure information, and that an important consequence of constructing an analyzer which would not only undertake deep as well as surface structure analysis but which would undertake both tasks simultaneously, would be that the number of incorrect surface structure parsings would be greatly reduced. The output of the present model shows that it is indeed the case. Again one notices that the result of designing the analyzer bearing in mind human behaviour (presumably in perceiving the syntactic structure of a sentence we do not first

perceive the surface structure and then work out the deep structure) is a considerable gain in its efficiency.

3. OUTLINE OF THE ANALYZER

The input consists of English sentences in a more or less normal orthographic form. The sentences are read and processed a word at a time and at the end of each sentence the analysis or analyses produced are displayed on the printer. In analyzing the sentences the program makes use of information about the syntactic functions of individual words derived from the closed-class dictionary and information about sentence structure derived from a representation of a grammar. At the start of a sentence the analyzer has a fixed set of initial predictions, and as it progresses through the sentence the existing predictions are tested, those which are satisfied are recorded, and new predictions are formed on the basis of the satisfied predictions and information about sentence structure obtained from the grammar. At the end of the sentence the record of satisfied predictions indicates all the possible analyses of the sentence with respect to the grammar. Because there is in principle no upper limit to the number of predictions which may arise in the course of analysis it is necessary to use some form of dynamic data structure to record the state of the predictions at successive points in the sentence. The analyses must also be recorded for subsequent output. In the surface-structure analyzer (as in other predictive analyzers) the prediction structure and the analysis structure were distinct. In the present model a single structure serves for both purposes. This may be viewed in two ways, either as an analysis record controlling the selection of further predictions from the grammar, or as a prediction structure in which fulfilled predictions are not discarded but are retained to become the record of the analysis. This method of implementation reflects quite literally the predictive principle; that is, that the way in which the later part of a sentence may be analyzed depends upon the analysis of the earlier part.

4. THE GRAMMAR

The grammar incorporated in the analyzer is a form of transformational grammar. A transformational grammar consists of a base component and a transformational component. The base component specifies a set of strings which correspond roughly to the simple or kernel sentences of the language and the transformational component accounts for complex sentences by deriving them from the basic underlying strings. The base component of the grammar associated with the analyzer here described differs in a number of respects from the type usually proposed. The three most significant differences are:

1. The base component is (structurally) a *regular grammar* rather than the customary *context-free phrase-structure grammar.*[1] A regular (or finite-state)

[1] For an account of regular grammars (expressions) in the context of automata theory see Kleene (1956).

grammar enumerates a set of strings without assigning any hierarchical structure to them. The adequacy of such a grammar as a base component depends in part on acceptance of the principle that all recursive constructions in a language necessarily involve transformations, since they cannot be generated by means of the base rules alone. In the case of English this principle has the possibly controversial consequence that all noun phrases have to be regarded as transforms because of the occurrence of phrases like *the old man's hat*, in which the relation of determiner is recursively realized by a possessive. (In fact, since possessives are an example of left-branching recursion, they are not beyond the weak generative capacity of a regular grammar; however, the genuinely hierarchical structure of the construction cannot be represented by such a grammar.) The total exclusion of phrase-structure rules from the grammar may be felt to be in some ways too strong a constraint. On the other hand, it means that we avoid some of the problems arising from the fact that in certain cases conventional phrase-structure grammars assign too much structure to base strings, with the result that statements of transformations become uneconomical.

2. The form of the grammar also allows for the specification of properties like number, case, tense, etc., in the form of lists of feature-values associated with the elements in the rules. This provides an apparatus for sub-categorization and for the application of rules for feature concord. The fact that subcategorial distinctions are represented in this form and not by the addition of extra categories results in economy both in the grammar and in the analysis procedure.

3. Explicit recognition is given in the grammar to the distinction between the concepts of syntactic form (involving such classificatory terms as *article*, *noun*, *nominal clause*, etc.) and syntactic function (involving such relational terms as *determiner*, *head*, *subject*, etc.). It is of course desirable that a syntactic analyzer for a natural language should not simply label the components of a sentence with category names but should also mark the relations which hold among the components, but it is also the case that markers of syntactic relations (hereafter SRMS) are more appropriate for the statement of certain generalizations such as those affecting feature concord.

Thus the base component is a regular grammar specifying a set of unstratified strings. The elements in the strings have three constituents: an SRM, a category name, and a list of feature values. The base component directly enumerates the prelexical strings for such simple sentences as *I like Sylvia*, *She visited him yesterday*, *He must have moved*, and so forth. In addition the grammar contains substitution rules and transformational rules proper. The substitution rules govern the realization of elements in the grammar–for the most part in a context-sensitive manner–either by lexical items or by transforms. The structural simplicity of the base makes it possible to regard the majority of the transformational rules as meta-rules in the sense that they operate on other rules to produce derived rules rather than operating on

structural descriptions to produce new structural descriptions. A practical consequence of regarding transformational rules in this way is that in many cases the effect of a transformational rule can be represented by a small number of derived rules of the same form as the base rules. Accordingly the grammar table (hereafter GT) actually used by the analysis program has the form of a finite-state network or directed graph–a form appropriate for the representation of a regular grammar–but it contains in addition to the base rules the derived rules for those transformations whose resultants are representable within this framework. The remaining transformational rules have to be applied dynamically in the course of the analysis procedure. It should be noted that the inclusion of SRMS and features in the grammar ensures that relevant syntactic information is preserved in the derived rules (e.g. selectional constraints persevere and basic relations remain marked).

4.1

The output of the analyzer reflects the form of the grammar. Analyses are displayed in the printout in a series of levels, the level structure reflecting the transformational structure of the sentence. The analysis produced for the simple sentence *She visited him yesterday* is

```
1  SE:STAT                                              TE:.  1
2  SU:she  AV:visited  OB:him  MO:yesterday                   2
```

(The letters preceding the colons are abbreviations for SRMS. SE stands for *sentence*, SU for *subject*, AV for *active verb*, OB for *object*, MO for *modifier*, and TE for *terminator*. See Appendix III for full details of the conventions used and for examples.) It will be noted that within the top-level phrase, STATEment, no structure has been assigned and the syntactic relations are shown as being realized directly by individual words. For the complex sentence *He asked who admired Descartes* the analysis is

```
1  SE:STAT                                              TE:.  1
2  SU:he  AV:asked  OB:INDQ                                   2
3                   SU:who  AV:admired  OB:Descartes          3
```

In this case the relation of object of the main clause is shown as being realized by a transform, INDirect Question.

The way in which deep-structure information is preserved, despite the considerable differences produced by different transformations in the form of identical deep-structure elements, is illustrated by the analysis for the sentence *Mary hates my teasing her.*

```
1  SE:STAT                                              TE:.  1
2  SU:Mary  AV:hates  OB:GER                                  2
3                     SU:my  AV:teasing  OB:her               3
```

In the interests of keeping the printout compact, the feature values associated with the components of the sentence are not reproduced in the output. Information about such facts as the relation of *who* to the pronouns *he* and *she* and the relation of *my* in the GERund to the pronoun *I* is contained in the dictionary, so that, if one wanted to, it would be possible to amend the program in such a way that the output relating to the deep structure would be in the form of kernel sentences, e.g. in the case of the last example *Mary hates it* and *I tease her.*

5. THE CLOSED-CLASS DICTIONARY

It has already been mentioned that the program does not need to have access to a complete dictionary giving the possible form-class assignments of every word. It operates with a list containing only certain classes of words. Some further consideration will now be given to the composition and use of this closed-class dictionary (CCD).

5.1

Three types of items are held in the CCD. First, there is a list of the grammatical formatives such as prepositions, pronouns, conjunctions, articles, and so on; secondly, there is a list of special verbs; and thirdly, there is a list of suffixes. Syntactic information about each word in the CCD is provided by one or more code words, which constitute the dictionary entry for the item. Each code word specifies a category or form class to which the item belongs and a list of values for the features associated with that category.

The relatively small class of items which belong to the first type mentioned above must be listed exhaustively in the CCD. This is necessary because words of this type provide essential information about the structure of any sentence.

The second list is of words which give information about certain kinds of sentence structure. For instance, it may be remarked that *Fred gave the dog biscuits* is ambiguous, whereas *Fred lost the dog biscuits* is not. This reflects the fact that the verbs *give* and *lose* have different properties. In particular, *give* may take two objects while *lose* may take only one object. We could have chosen to treat all open-class words as potential double-object verbs, which would result in an unacceptable analysis being produced for the second sentence. Instead it seems preferable to say that open-class words, when treated as verbs, can take at most one object, and to list verbs like *give* in the CCD so that two correct analyses are produced for the first sentence and only one for the second. Similarly, certain other classes of verbs are listed in the CCD; for instance, those which take a complement instead of an object (*He looked a fool*), and those which take an object and an infinitive (*He made her cry*).

The suffixes contained in the CCD are restricted to inflections. Examples of the endings included are *-s*, *-ed*, *-ing*, *-'s*, and *-s'*. Like the grammatical formatives, these elements carry essential information about sentence structure. The dictionary look-up procedure in the program automatically recognizes

these endings and detaches them from the stem of the word. Simple checks are incorporated to ensure that words such as *bed* and *bus* are not treated as if they ended with the inflectional *-ed* or *-s*. Other exceptions, either individual words (e.g. *news*, *lens*) or non-inflectional ending classes (e.g. *-ss*, *-ous*), must be listed in the CCD. Conversely, words for which the information normally carried by the suffixes is specified in a non-standard way (e.g. the past tense forms of strong verbs) must also be listed as exceptions.

A fuller account of the contents of the CCD has been given elsewhere (Bratley and Dakin, 1968). It is worth emphasizing again that, compared to a complete dictionary of English, the CCD is very short. The list of grammatical formatives contains a few hundred items, and while the lists of words of the second type have not been fully enumerated at the present time, the total number of items in the dictionary should not exceed 2000.

5.2

The CCD is stored as a tree structure so that, for instance, the four words *this*, *the*, *them*, and *to* would be represented in the form shown in figure 1, where

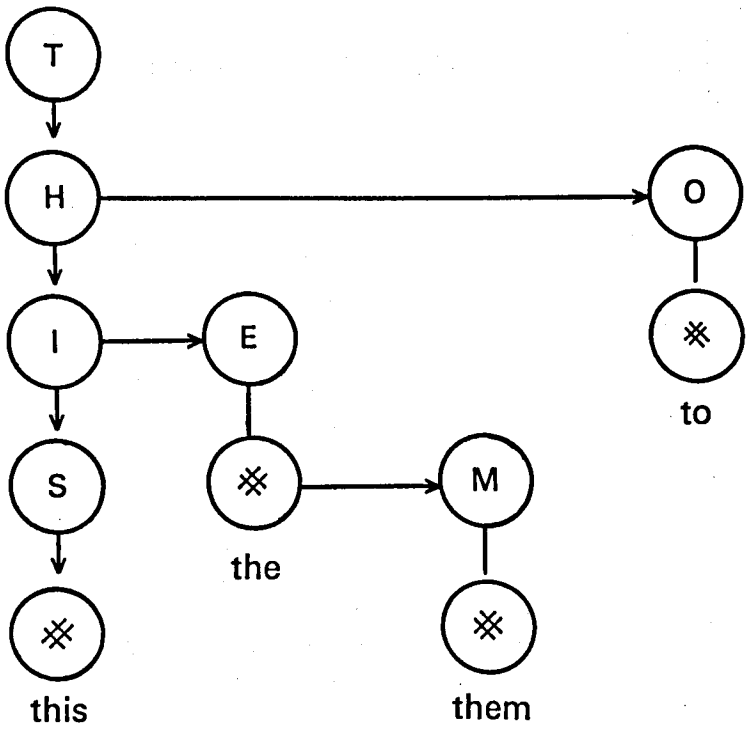

Figure 1

(i) ※ , representing 'end-of-word', is treated as an extra letter;
(ii) downward links represent progression through the successive letters of a word;
(iii) cross-wise links represent an alternative choice of letter at the same position in a word;
(iv) there are pointers from the 'end-of-word' nodes to the relevant dictionary entries.

The endings are stored in a similar tree structure, and the use of default links from the main structure enables inflected words to be recognized in almost exactly the same way as listed words.

Special entries are contained in the dictionary for open-class words, proper names, and numerals. Since the information carried by an inflected word may not be categorical but may be a function of the stem of the word as well as the suffix, provision is made for including with each item an operation code which is held in the 'end-of-word' node. This code specifies an operation to be performed on the feature values entered for the stem of the word or, if the stem is not listed, on the entry for open-class words. The use of operation codes is not in fact confined to the ending-classes; they may also be used to indicate the relationship of an individual word to another item. The following fragment of the CCD in the form in which it is submitted to the set-up routine illustrates the kind of facilities provided. (In the interests of simplifying the set-up procedure, feature values are written as numbers corresponding to the required binary patterns.)

```
      -s = -(23)
     -ed = -(24)
    -ing = -(25)
     -ss = —
    the : art    0 1 0 3 0
   a,an : art    0 1 0 1 0
can,must : modl  0 1 7 8 0
        : noun   0 1 3 1 0
   they : pro    0 1 1 2 1
  their = they (1)
   know : verb   0 3 6 8 50
   knew = know (10)
  known = knew (9)
 fought = -(4)
```

6. THE ANALYSIS PROCEDURE

The task of the analysis procedure is essentially the progressive construction of a data structure in which the predictions realized by successive words of the sentence to be analyzed are recorded and which is then used to determine what new predictions may be made for the following words. The procedure has available to it the stored GT and the dictionary look-up routine which

assigns codings to each input word in the manner described in the preceding section. The analysis structure produced is basically a diverging tree with nodes which may themselves represent sub-trees. At the start of the operation the structure consists of a single node which represents the root of the main tree.

For each word of the sentence the 'open ends' of the structure (i.e. the latest active nodes so far added on each analysis path) are traversed and the predictions specified by them are tested. In general, the immediate predictions which may be made from a node are determined by means of a reference to an element in the GT, the successors of this element in the finite-state network representing the possible continuations of the analysis path, but in a number of cases, some of which are mentioned below, they are computed from information held in the analysis structure. For a prediction of an element which is realizable directly by a lexical item, the codings of the input word are matched against the predicted values and if the match is successful a new simple node is added to the analysis path. In the case of a prediction realizable by a transform phrase, reference is made to a table (TPT) to determine if this is the first prediction of the phrase encountered for the current word. If it is, a new tree is established with a root node specifying the initial predictions for the phrase and these predictions are in turn investigated. If it turns out that none of the initial predictions is successful, a failure indicator is set in the TPT, otherwise a node representing the new sub-tree is added to the analysis path from which the original prediction was made. Where a transform phrase prediction is not the first encountered, it is necessary only to perform the last-mentioned step–unless the failure indicator is set in the TPT, in which case no action is taken. Thus multiple predictions of the same type of phrase give rise to the establishment of a single sub-tree.

If a node terminates a complete analysis of a transform phrase, the overall feature values for the phrase analysis are computed and recorded so that subsequently the higher-level paths on which the phrase prediction was encountered can be reactivated. The reactivation does not occur until all complete analyses of the phrase ending on the same word have been recognized, so that a single reactivation can be made with feature values which represent all the analyses. For example, one possible analysis of the noun phrase *the research computing needs*, in which *computing needs* is taken as a relative clause, makes it singular, while the other makes it plural; the value for the feature of number that is required for the whole phrase is therefore singular-or-plural. The feature values for a phrase are matched against the predicted values on the higher-level paths in the same way as for lexical items.

Certain types of node (e.g. the root nodes of sub-trees) are created provisionally and do not remain on the structure unless they eventually lead to a successful analysis of the current word, but otherwise the structure grows progressively and analysis paths which peter out are not removed, so that, in fact, at the end of the sentence the analysis structure represents not only all complete analyses but all partial analyses as well. However, the recognition of

identical phrase predictions–which prevents ambiguity at a higher level from being extended to a lower level–and the single representation of multiple analyses of a phrase–which prevents ambiguity at a lower level from being extended to a higher level–serve to keep the size of the analysis structure and the amount of processing required within reasonable bounds (see figure 2).

When the end of a sentence has been reached, all complete analyses recorded on the analysis structure are traced and printed out. If at any point before the end of the sentence no prediction remains, this indicates that the grammar does not provide any analysis for the sentence (either because the grammar is incomplete or because the sentence is in fact ungrammatical), and the comment *no complete analyses* is printed out.

It was indicated in the section describing the grammar associated with the analyzer that while the resultants of most transformational rules are incorporated in the GT, the effect of some cannot be completely specified in this way and must be developed in the course of the analysis process. The two main instances of this relate to constructions involving inversion and constructions involving co-ordination.

Inversion is exemplified in such interrogative and relative constructions as

Is he bringing his wife? (inversion of auxiliary)
the book which he bought (inversion of object of *bought*)
the boy who she said kicked her (inversion of subject of *kicked*)
Which hat will she buy? (inversion of both auxiliary and object)

The rules for these constructions are formulated in such a way that the inverted words or phrases are accepted at the point at which they occur in the sentence (that is, their surface structure position), but with the assignment of a dummy SRM (OO) indicating that in general it is not possible at this point to specify what syntactic relation is exemplified by the element. All subsequent nodes added to an analysis path involving an inverted element include a link back to it. The effect of this is to make the item available for fulfilling a subsequent prediction, so that in these cases a prediction is satisfied by a null input (that is, without using up any words of the sentence). The realization of a prediction in this way is marked in the output by an asterisk following the SRM. Thus for the examples listed above the analysis printout would be:

```
1  SE:QUES                                                    TE:?  1
2  OO:is  SU:he  AU:*  AV:bringing  OB:CNP                          2
3                                   DE:his  HE:wife                 3

DE:the  HE:book  AT:REL
                 OO:which  SU:he  AV:purchased  OB:*

DE:the  HE:boy  AT:REL
                OO:who  SU:she  AV:said  OB:INDS
                                         SU:*  AV:kicked  OB:her
```

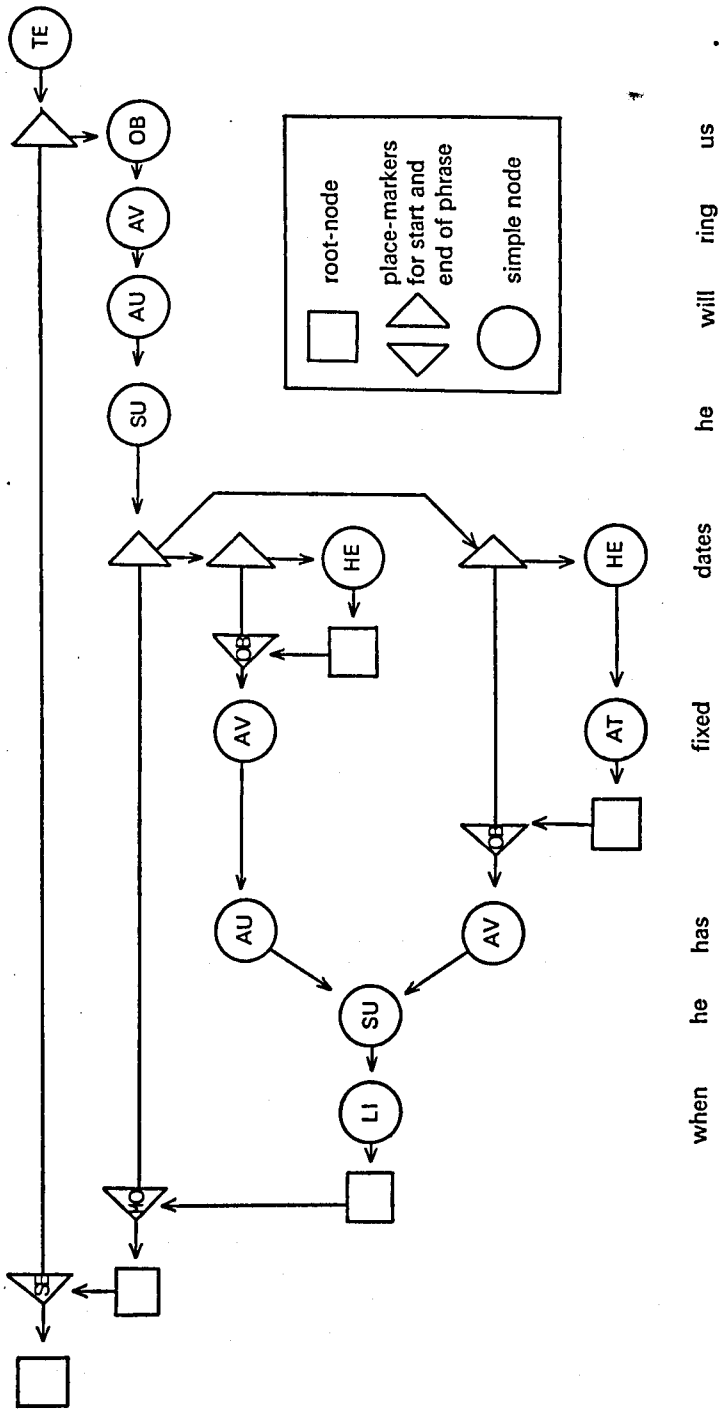

Figure 2. Illustration of analysis structure for sentence 22 (showing only those nodes which figure in complete analyses)

```
1  SE:QUES                                                    TE:?  1
2  OO:CNP          OO:will  SU:she  AU:*  AV:buy  OB:*               2
3  DE:which  HE:hat                                                 3
```

An asterisk following an SRM more generally indicates the deletion of an element, as for example in imperative sentences like *Go*, where it indicates the non-realization of the deep structure subject.

```
1  SE:IMP              TE:.                                         1
2  SU:*  AV:go                                                      2
```

Similarly in sentences like *I met a man I know*, the combination of the two conventions (OO:*) is used to indicate the absence of the relative pronoun. As with the full form of the relative clause, the deep structure position is also indicated (in this case by OB:*).

```
1  SE:STAT                                                    TE:.  1
2  SU:I  AV:met  OB:CNP                                             2
3                DE:a  HE:man  AT:REL                               3
4                              OO:*  SU:I  AV:know  OB:*            4
```

The procedure for dealing with co-ordination (constructions involving *and*, *or*, etc.) consists essentially in the reinstatement, following the co-ordinator, of predictions which have been made at an earlier point. Co-ordinators are not in fact predicted, and when they are encountered they are accepted without reference to the GT. The nodes created for these words record previous predictions which may now be reinstated, all subsequent nodes added to the analysis paths being marked with a co-ordination link (cf. the treatment of inversion). This link indicates that at some point the two limbs of the co-ordination must be 'brought together', i.e. analysis of the part of the sentence following the co-ordinator word must reach the same point as that reached immediately before it, so that a prediction common to both is satisfied.

In the output the point from which predictions were reinstated is marked with a left bracket and the point at which a successful common prediction was satisfied is marked by a right bracket. The brackets thus indicate the scope of the co-ordination. The analysis of the sentences *She danced and sang* and *Have John and his sister arrived?* is shown below. (For further examples see Appendix III.)

```
1  SE:STAT                          TE:.                          1
2  SU:she  (AV:danced and AV:sang)                                2
```

```
1  SE:QUES                                                    TE:?  1
2  OO:have (SU:John and SU:CNP          ) AU:* AV:arrived           2
3                    DE:his HE:sister                               3
```

Some of the problems of dealing with this type of construction are rather intractable. In particular, it is difficult to devise suitable conditions regarding the degree of similarity which has to be exhibited by the two limbs of a co-ordination. On the one hand it is possible to impose the very strict constraint that only predictions which have been *satisfied* in the first limb may be re-instated for the second, thus requiring identity of analysis for the two limbs. However, this would preclude the successful analysis of such entirely acceptable sentences as *Is he or is he not coming?* and *He asked for and was given a glass of water.* On the other hand, if the principle is adopted of permitting the reinstatement of any prediction which has been *made* (but not necessarily satisfied) in the first limb, there is a danger of accepting such examples of syntactic syllepsis as **He said his prayers and that he died a happy man.* In fact we have adopted a principle which is nearer the second of those cited above but which incorporates requirements of feature concord that effectively exclude at least the more obviously undesirable cases. The ordering of transformations associated with inversion and co-ordination relative to each other is also problematic. The two (apparently equivalent) analyses produced for sentence 28 reflect this difficulty. The problem here is not simply to eliminate one analysis, but to discover criteria for determining which this should be.

7. CONCLUSION

We have shown that it is possible to construct an effective syntactic analyzer operating under the constraints listed in Section 2. As one would expect, it has certain limitations.

1. A result of not using a full dictionary of English is, of course, that in the case of some sentences incorrect analyses are produced as well as the correct analysis. For example, in the case of the sentence *The cat adores fish*, as well as the analysis in which *adores* is taken as a verb, an analysis is produced in which *adores* is taken as a noun. (Notice that in the case of a sentence like *The girl guides fish* one would require these two analyses.) It should be emphasized that only a small number of incorrect analyses are attributable simply to a lack of form class information. It follows from this that in order to obtain a substantial improvement by supplying the program with a complete dictionary it would be necessary for it to provide more than merely form class information. It would, in fact, need to contain information about other features, such as, in the case of nouns, whether they are abstract or concrete, animate or inanimate, etc. A possible extension of the program which would enable it to derive information about such features automatically is discussed in Appendix I.

2. The program is not a general-purpose analyzer for arbitrary transformational grammars. It was pointed out in Section 4 that in order to give effect to certain transformational rules specific procedures had to be incorporated into the program. Given the present formulation of transformational rules in linguistic theory, it is doubtful whether a generalized algorithm is possible.

3. The attempt to present a perspicuous representation relating deep structure information to the actual surface structure of the sentence has led in some cases to a certain ambivalence–for example, the assignment of SRMS to grammatical formatives as if they were lexical elements rather than realizations of bundles of syntactic features.

4. At the present time the analyzer takes no account of *derivational affixes*. As a result it is unable to recognize nominalizations like *declaration, performance, management*, etc., and relate them to their underlying sentential forms.

5. There are deficiences in the treatment of certain kinds of constructions. Some of those affecting the analysis of sentences containing conjunctions like *and* have already been noted in Section 6. Similar problems (in an even more acute form) arise in the case of sentences containing words like *as* and *than*. In fact such sentences are outside the range of the analyzer. The reason for this –the partial failure to handle *and* and the total failure to handle *as* and *than*– is very simple. It is directly related to the fact that we ourselves have an incomplete understanding of the grammar of *and* and only vague ideas about the grammar of *as* and *than*. Only when developments in linguistic theory have resulted in a formalism capable of explicating the structure of these sentences will it be reasonable to expect an automatic analyzer to produce adequate analyses for sentences of this kind. It is necessary to make this obvious point in view of the many claims made in recent years to the effect that the development of an automatic syntactic analyzer will in itself help in solving these kinds of linguistic problems.

REFERENCES

Bratley, P. & Dakin, J. (1968), A limited dictionary for syntactic analysis. *Machine Intelligence 2*, pp. 173–81 (eds Dale, E. and Michie, D.). Edinburgh: Oliver and Boyd.

Chomsky, N. (1965), *Aspects of the theory of syntax*. Cambridge, Mass.: MIT Press.

Kay, M. (1967), Experiments with a powerful parser. *Deuxième conférence internationale sur le traitement automatique des langues*. Grenoble.

Kleene, S.C. (1965), Representation of events in nerve nets and finite automata. *Automata Studies*. Princeton.

Petrick, S.R. (1965), *A recognition procedure for transformational grammars*. Ph.D. thesis, MIT.

Thorne, J.P., Dewar, H., Whitfield, H. & Bratley, P. (1966), A model for the perception of syntactic structure. *Colloque international sur l'informatique*. Toulouse.

Zwicky, A.M., Friedman, J., Hall, B.C. & Walker, D.E. (1965), The MITRE syntactic analysis procedure for transformational grammars. *AFIPS*, 27. Fall J.C.C. Washington D.C.: Spartan Books.

APPENDIX I: A POSSIBLE EXTENSION OF THE PROGRAM

There is an interesting extension that could be made to the program. For the reasons given, words like *boy* and *laugh* are not entered in the dictionary employed in the analysis procedure, which means that no information for them is obtained from the dictionary. But using information derived from the analyses it produces the program could, so to speak, 'learn' that *boy* is a noun and *laugh* a verb, and could construct for itself another dictionary–an open-class dictionary–in which this information would be stored.

Notice that in many cases during the early stages of running the program it is inevitable that incorrect entries would be made in the open-class dictionary. Given the sentence *The cat adores fish*, the analyzer produces two analyses–the desired analysis and one in which *The cat adores* is taken as a noun phrase on the analogy of phrases like *the boy scouts*. *Adores* would therefore be entered tentatively both as a verb and a noun. But it seems reasonable to suggest that after a while an automatic correction routine could be run on the open-class dictionary which, for example, would discover any word entered as being both a noun and a verb but for which, while there have been unambiguous instances of its being labelled as a verb, no cases have been found in which it has been labelled as a noun without, at the same time, an analysis being produced in which it has been labelled as a verb. In this case the dictionary entry for the word would be modified by the deletion of the label *noun*. If at a later stage the same sentence were submitted for analysis, two analyses would again be produced. But now both analyses could be checked against the open-class dictionary the program has itself constructed. If taking the sentence as ambiguous meant treating one of the words as a part of speech different from that the dictionary records it as belonging to, then this would be a sufficient reason for dropping this analysis. Given an analysis of a sentence in which every word functions as the part of speech as which it usually functions, we are unlikely also to accept another analysis for the sentence in which one of the words now functions in an entirely unexpected way. For example, no one is likely to take the sentence *Power corrupts* as an imperative, on the analogy of a sentence like *Bring water*, because this would involve taking *corrupts* as a noun and there is a perfectly acceptable analysis of the sentence in which *corrupts* functions in the expected way, as a verb.

Following out this procedure it would be possible for the program not only to acquire the information that *boy* is a noun and *laugh* a verb, but also the information that *laugh* is an intransitive verb. But if the program is to acquire all the information the English speaker has about these words, it is also necessary that it should contain the information that *boy*, for example, is a concrete noun and an animate noun. It is possible that this kind of information too might be automatically derived. For this to happen, however, it would first be necessary for information about such properties to be supplied for certain words. Say, for example, we were to include in the original closed-class

dictionary the word *surprise* plus the information that it is a verb that must always take an animate noun as its object. Then, when the sentence *The boy surprised the teacher* has been analyzed, the program will have learnt not only that *teacher* is a noun but also that it is an animate noun. If the next sentence to be analyzed is *The teacher laughed*, it would now have learnt that *laugh* is the kind of verb that can take an animate subject. In this way it seems that the original information concerning the properties of a few words could be spread over the whole lexicon.

Making the analytic procedure and the open-class dictionary arising from it operate together in this way would have the following effect: as the open-class dictionary using the information produced by the analyzer improved so too would the analyses produced. However, there are many problems here. Many verbs can take animate, inanimate and abstract subjects, and the fact that up to a certain point the program has not encountered an instance of a verb taking one type of subject is no guarantee that it will not do so. Moreover, it is by no means clear which verbs, or how many verbs, or even whether verbs, should be chosen as the starting point. Nevertheless, this looks as though it would be an interesting field for experiment.

APPENDIX II: PROGRAMMING DETAILS

The analysis program runs at present on the KDF9 computer at Edinburgh, which has 16K 48-bit words of core store. Core cycle time is about 4 to 6 microseconds, depending on overlapping, and simple fixed-point instructions take about 2 microseconds to execute in the high-speed registers. The machine-code instruction format is variable length, so that on average two or three instructions can be held in one computer word. Of the 16K words of core available, about 2K are taken up by the supervisor and operating system, about 4K are occupied by the instructions of the program, and the rest are available for data space. In addition to the analysis routines the program contains routines to accept and set up the grammar and dictionary in the required internal format. The fact that the whole CCD is small enough to be held in main store naturally results in a considerable economy in the overall processing time for a sentence. It is convenient to hold the compiled program, including the grammar and dictionary, on magnetic tape, but the program itself requires no backing store.

The program is written in Atlas Autocode, a high-level language akin to ALGOL. This provides the essential features of recursive procedure calling and the specification of parameters either by name or value, as well as a number of simple facilities for handling non-numeric data. As the version of the language currently available does not include provision for Boolean and shift operations, a few small procedures for manipulating operands have had to be programmed in machine code. The program currently runs to about 1000 lines of Atlas Autocode, equivalent to some ten thousand instructions after compilation.

Sentences to be analyzed are submitted in free format on paper tape produced by a Flexowriter. At present the input is subject to the restrictions that only proper names and the pronoun *I* may start with a capital letter and that abbreviations terminated by a period may not be used.

APPENDIX III: EXAMPLES OF OUTPUT

The computing time for each sentence and the total number of nodes in the analysis structure are recorded on the line following the sentence. The timing figure (in seconds to three decimal places) includes the time spent in consulting the CCD; it does not include the time spent in outputting the analyses. The arrangement of the printout is hierarchical, with the elements entering into the analysis of each category being printed out under the category name.

The abbreviations used are listed below.

SRMS (*syntactic relation markers*)	
[SE	sentence]
TE	terminator
SU	subject
AV	active verb
OB	object
MO	modifier
AU	auxiliary
DE	determiner
HE	head (of noun phrase)
AT	attribute
IN	indirect object
LI	link (preposition or conjunction)
PO	prepositional object
PA	particle
CO	complement
OO	INVERTED ELEMENT

Category names	
STAT	statement
QUES	question
IMP	imperative
INDS	indirect statement
INDQ	indirect question
INFC	infinitive clause
NOMC	nominal clause
PARC	participial clause
SUBC	subordinate clause
GER	gerund
REL	relative
PREC	prepositional clause
CNP	complex noun phrase

1 she visited him yesterday.
(time taken: 0·565 seconds nodes used: 8)

```
1 SE:STAT                                  TE:. 1
2 SU:she AV:visited OB:him MO:yesterday       2
```

2 he must have moved.
(time taken: 0·557 seconds nodes used: 12)

```
1 SE:STAT                                TE:. 1
2 SU:he AU:must AU:have AV:moved            2
```


3 the cat adores fish.
(time taken: 1·044 seconds nodes used: 32)

```
1 SE:STAT                                TE:. 1
2 SU:CNP          AV:adores OB:CNP          2
3 DE:the HE:cat             HE:fish         3
```

```
1 SE:STAT                                TE:. 1
2 SU:CNP                  AV:fish           2
3 DE:the AT:cat HE:adores                   3
```

4 this cat adores fish.
(time taken: 0·978 seconds nodes used: 28)

```
1 SE:STAT                                TE:. 1
2 SU:CNP          AV:adores OB:CNP          2
3 DE:this HE:cat            HE:fish         3
```

5 *go.*
(time taken: 0·427 seconds nodes used: 11)

```
1  SE:IMP          TE:.  1
2  SU:*  AV:go           2
```

6 Mary hates my teasing her.
(time taken: 1·035 seconds nodes used: 29)

```
1  SE:STAT                                          TE:.  1
2  SU:Mary  AV:hates  OB:GER                              2
3                     SU:my  AV:teasing  OB:her           3
```

7 the boy who kissed the girl laughed uproariously.
(time taken: 1·281 seconds nodes used: 36)

```
1  SE:STAT                                                          TE:.  1
2  SU:CNP                                    AV:laughed  MO:uproariously   2
3  DE:the  HE:boy  AT:REL                                                  3
4                  SU:who  AV:kissed  OB:CNP                               4
5                                     DE:the  HE:girl                      5
```

8 the boy who the girl kissed laughed uproariously.
(time taken: 1·294 seconds nodes used: 38)

```
1  SE:STAT                                                          TE:.  1
2  SU:CNP                                    AV:laughed  MO:uproariously   2
3  DE:the  HE:boy  AT:REL                                                  3
4                  OO:who  SU:CNP          AV:kissed  OB:*                 4
5                          DE:the  HE:girl                                 5
```

9 the boy the girl kissed laughed uproariously.
(time taken: 1·176 seconds nodes used: 37)

```
1  SE:STAT                                                                  TE:.  1
2  SU:CNP                                                  AV:laughed  MO:uproariously   2
3  DE:the  HE:boy  AT:REL                                                               3
4                  OO:*  SU:CNP            AV:kissed  OB:*                              4
5                        DE:the  HE:girl                                                5
```

10 Fred gave the dog biscuits.
(time taken: 0·928 seconds nodes used: 23)

```
1  SE:STAT                                          TE:.  1
2  SU:Fred  AV:gave  OB:CNP                               2
3                    DE:the  AT:dog  HE:biscuits          3
```

```
1  SE:STAT                                          TE:.  1
2  SU:Fred  AV:gave  IN:CNP          OB:CNP               2
3                    DE:the  HE:dog  HE:biscuits          3
```

11 Fred lost the dog biscuits.
(time taken: 0·863 seconds nodes used: 19)

```
1  SE:STAT                                          TE:.  1
2  SU:Fred  AV:lost  OB:CNP                               2
3                    DE:the  AT:dog  HE:biscuits          3
```

12 pick a ripe banana.
(time taken: 0·879 seconds nodes used: 26)

```
1  SE:IMP                                       TE:.  1
2  SU:*  AV:pick  OB:CNP                              2
3                 DE:a  AT:ripe  HE:banana            3
```

13 *when did John say he would come?*
(time taken: 1·294 seconds nodes used: 30)

```
1  SE:QUES                                                          TE:?  1
2  OO:when  OO:did  SU:John  AU:*  AV:say  MO:*  OB:INDS                 2
3                                                 SU:he  AU:would  AV:come  3

1  SE:QUES                                                          TE:?  1
2  OO:when  OO:did  SU:John  AU:*  AV:say  OB:INDS                        2
3                                       SU:he  AU:would  AV:come  MO:*    3
```

14 he observed the man with the telescope.
(time taken: 1·061 seconds nodes used: 26)

```
1  SE:STAT                                                          TE:.  1
2  SU:he  AV:observed  OB:CNP                                             2
3                      DE:the  HE:man  MO:PREC                            3
4                                      LI:with  PO:CNP                    4
5                                               DE:the  HE:telescope      5

1  SE:STAT                                                          TE:.  1
2  SU:he  AV:observed  OB:CNP          MO:PREC                            2
3                      DE:the  HE:man  LI:with  PO:CNP                    3
4                                               DE:the  HE:telescope      4
```

15 power corrupts.
(time taken: 0·724 seconds nodes used: 20)

```
1  SE:STAT                TE:.  1
2  SU:CNP   AV:corrupts         2
3  HE:power                     3
```

```
 1  SE:IMP                          TE:.  1
 2  SU:*  AV:power  OB:CNP                2
 3                  HE:corrupts           3
```

16 he is waiting.
(time taken: 0·535 seconds nodes used: 12)

```
 1  SE:STAT                  TE:.  1
 2  SU:he  AV:is  CO:waiting       2
```

```
 1  SE:STAT                  TE:.  1
 2  SU:he  AU:is  AV:waiting       2
```

17 playing cards intrigues me.
(time taken: 1·045 seconds nodes used: 28)

```
 1  SE:STAT                                          TE:.  1
 2  SU:GER                    AV:intrigues  OB:me          2
 3  SU:*  AV:playing  OB:CNP                               3
 4                    HE:cards                             4
```

18 playing cards intrigue me.
(time taken: 1·026 seconds nodes used: 29)

```
 1  SE:STAT                                  TE:.  1
 2  SU:CNP               AV:intrigue  OB:me        2
 3  AT:playing  HE:cards                           3
```

19 I dislike playing cards.
(time taken: 0·951 seconds nodes used: 21)

```
 1  SE:STAT                              TE:.  1
 2  SU:I  AV:dislike  OB:CNP                   2
 3                    AT:playing  HE:cards     3
```

```
1  SE:STAT                                                    TE:.  1
2  SU:I  AV:dislike  OB:GER                                         2
3                    SU:*  AV:playing  OB:CNP                       3
4                                      HE:cards                     4
```

20 the rascal who John claimed committed the crime has escaped.
(time taken: 1·840 seconds nodes used: 44)

```
1 SE:STAT                                                                          TE:. 1
2 SU:CNP                                                          AU:has AV:escaped    2
3 DE:the HE:rascal AT:REL                                                            3
4                  OO:who SU:John AV:claimed OB:INDS                                 4
5                                            SU:* AV:committed OB:CNP              5
6                                                                DE:the HE:crime     6
```

21 whose book did you say you wanted?
(time taken: 1·537 seconds nodes used: 36)

```
1  SE:QUES                                                                  TE:?  1
2  OO:CNP              OO:did  SU:you  AU:*  AV:say  OB:INDS                       2
3  DE:whose  HE:book                         SU:you  AV:wanted  OB:*              3
```

22 when he has fixed dates he will ring us.
(time taken: 2·285 seconds nodes used: 60)

```
1  SE:STAT                                                          TE:.  1
2  MO:SUBC                                  SU:he  AU:will  AV:ring  OB:us  2
3  LI:when  SU:he  AV:has  OB:CNP                                           3
4                          AT:fixed  HE:dates                               4
```

```
1  SE:STAT                                                          TE:.  1
2  MO:SUBC                                  SU:he  AU:will  AV:ring  OB:us  2
3  LI:when  SU:he  AU:has  AV:fixed  OB:CNP                                 3
4                                    HE:dates                             4
```

23 the queen's sister's husband took good photographs.
(time taken: 1·521 seconds nodes used: 45)

```
1  SE:STAT                                                                  TE:.  1
2  SU:CNP                                          AV:took  OB:CNP                2
3  DE:CNP                        HE:husband                AT:good  HE:photographs  3
4  DE:CNP              HE:sister's                                               4
5  DE:the  HE:queen's                                                            5
```

24 a lawyer who cheats the clients he sees deserves censure.
(time taken: 1·863 seconds nodes used: 47)

```
1  SE:STAT                                                                                  TE:.  1
2  SU:CNP                                                                AV:deserves  OB:CNP      2
3  DE:a  HE:lawyer  AT:REL                                                          HE:censure    3
4                SU:who  AV:cheats  OB:CNP                                                       4
5                          DE:the  HE:clients  AT:REL                                            5
6                                          OO:*  SU:he  AV:sees  OB:*                           6
```


25 has the portrait they bought disappeared?
(time taken: 1·001 seconds nodes used: 22)

```
1  SE:QUES                                                          TE:?  1
2  OO:has  SU:CNP                                  AU:*  AV:disappeared   2
3          DE:the  HE:portrait  AT:REL                                    3
4                           OO:*  SU:they  AV:bought  OB:*                4
```

26 he rolled up the bright red carpet.
(time taken: 1·363 seconds nodes used: 42)

```
1  SE:STAT                                                  TE:.  1
2  SU:he  AV:rolled  MO:PREC                                      2
3                    LI:up  PO:CNP                                3
4                           DE:the  AT:bright  AT:red  HE:carpet  4
```

```
1  SE:STAT                                                    TE:.  1
2  SU:he  AV:rolled  PA:up  OB:CNP                                  2
3                           DE:the  AT:bright  AT:red  HE:carpet    3
```

27 she handed John a pear and Mary an apple.
(time taken: 1·322 seconds nodes used: 34)

```
1  SE:STAT                                                          TE:.  1
2  SU:she  AV:handed  (IN:John  OB:CNP        and  IN:Mary  OB:CNP       )   2
3                           DE:a  HE:pear                DE:an  HE:apple    3
```

28 the plants he watered and tended flourished.
(time taken: 1·480 seconds nodes used: 45)

```
1  SE:STAT                                                          TE:.  1
2  SU:CNP                                                AV:flourished    2
3  DE:the  HE:plants  AT:REL                                              3
4                     OO:*  SU:he  (AV:watered and AV:tended)  OB:*       4
```



```
1  SE:STAT                                                          TE:.  1
2  SU:CNP                                                AV:flourished    2
3  DE:the  HE:plants  AT:REL                                              3
4                     OO:*  SU:he  (AV:watered  OB:*  and  AV:tended  OB:*)  4
```

29 are the elephant and the kangaroo he adopted obeying him?
(time taken: 1·661 seconds nodes used: 49)

```
1 SE:QUES                                                          TE:? 1
2 OO:are SU:CNP                                    AU:* AV:obeying OB:him  2
3        (DE:the HE:elephant and DE:the HE:kangaroo) AT:REL                3
4                                  OO:* SU:he AV:adopted OB:*              4
```

```
1 SE:QUES                                                          TE:? 1
2 OO:are SU:CNP                                    AU:* AV:obeying OB:him  2
3        (DE:the HE:elephant and DE:the HE:kangaroo AT:REL          )      3
4                                  OO:* SU:he AV:adopted OB:*              4
```