[*International Conference on Machine Translation of Languages and Applied Language Analysis*, National Physical Laboratory, Teddington, UK, 5-8 September 1961]

A NEW MODEL OP SYNTACTIC DESCRIPTION

by

A.F. PARKER-RHODES

(Cambridge Language Research Unit, England)

PREFACE

THIS paper expounds the lattice theory of syntax developed by the Cambridge Language Research Unit during the last five years. The idea, that the conceptual apparatus of lattice theory could be put to use in the description of linguistic phenomena originated with M. Masterman in 1956, and was first put before the public in a paper by A.F. Parker-Rhodes, read at a Machine Translation colloquium organized by M.I.T. in Dedham, Mass. in that year. At this stage, the device described here as the "meet algorithm" was the only part of the theory that had been clearly formulated; but the later developments were already in some measure foreseen. Since that occasion, this is the first formal presentation of the theory to be published.

A fuller account of the theory, together with a detailed account of the program based on it, is to be brought out shortly, in the following four parts:

I. The Lattice Properties of Syntactic Relations in an Open Language.	A.F. Parker-Rhodes
II. Derivation of Syntactic Relations from a Lattice Model	A.F. Parker-Rhodes M. Masterman
III. Relation between the Theory and its Application to Syntax Analysis Programming.	K.S. Jones
IV. The CLRU Syntax Analysis Program	A.F. Parker-Rhodes R. McKinnon-Wood M. Kay. P. Bratley

1. INTRODUCTION

This paper describes briefly a new model of grammatical description, devised originally with the purpose of providing a better tool for the machine processing of language material. Particular attention has been given to the advantages likely to accrue, for this purpose, from exploiting to the full whatever features could be found in common between all languages. The need to devise a new model became apparent when it was found how little attention had been given in the past to this point.

It seems that previous models of grammatical description fall into four main classes. The oldest of these, which has been called by Hockett (6) the "Word-and-Paradigm" or WP model, originated in antiquity, and is well adapted to the description of inflected languages like Sanskrit, Greek and Latin. It is however, despite Robins' (3) recent reconsideration, far too limited in scope for our purposes. The next, the "Item-and-Process" or IP model in Hockett's terminology, works with the notion of items (word or short phrases) being modified by various processes (suffixation, vowel-change, root-replacement, etc.) to produce all the various forms of the language. This model was first clearly systematized by Sapir (9); it is more adaptable than the WP model, but still not sufficiently general. The "Item-and-Arrangement" or LA model was evolved by descriptive linguists; it aims to describe the whole grammar of a language in terms of lists of items and of the ways in which they can be arranged (i.e. constructions). This model lends itself better to expressing the basic hierarchical structure of sentences, first recognised clearly by Husserl (7), than the previous models, and is somewhat easier to formulate mathematically; but it runs into numerous difficulties which have led to the formulation of yet another type of model.

This is the one originated by Harris (4) and greatly strengthened by Chomsky (2); we may call it the Kernel-and-Transformation or KT model. It takes as its starting point, a number of simple standard sentence forms, called "kernels", and seeks to derive every possible correct sentence in the language by developing these kernels through a mechanism of substitution of their components by other kernels. This model has a number of advantages, notably in the description of what I here call interrupted substituents, but it is very refractory to mathematical formulation. This model has received a more extensive application to problems of handling language material and mechanization of language processes than the others. This work is especially associated with the University of Pennsylvania, where it has been ingeniously used by Hiz (5) and by Kaufman (8). Unfortunately the great complexity produced by these efforts, even though they have been confined

to the description of a single language (English) casts some doubt on the effectiveness of the KT model for our purposes.

The new model which I propose here, for the purpose of meeting the needs of machine translation better than those previously have done, will be set out so far as possible in an axiomatic manner, in order to emphasize its internal structure. The task of demonstrating in detail its application to the description of actual languages lies outside the scope of this paper. Evidence that it is so applicable comes from two sources: first, the operation of machine programs embodying ideas drawn from the model for the syntactic analysis of texts; and second, descriptions of various particular languages capable of being compared with each other and with more conventional descriptions. Evidence of both sorts is planned for publication in due course; here, I shall confine myself to exposition alone.

First, I shall define an operation called "replacement" by which parts of utterances may be may be substituted by other parts: this does no more than re-state familiar Ideas. Second, I shall use this operation to derive a rigorous definition of grammatical function (in a partly mathematical context this term, unfortunately, is too liable to be misunderstood, and must be replaced: I use the term "paradigm", in an analogically extended sense, for this purpose). Third, I show that the set of all possible paradigms (functions) constitutes a well-defined mathematical system, namely, a lattice; this makes possible major simplifications in the description of syntactic phenomena. Fourth, I shall use the conceptual apparatus to hand to circumscribe the possible diversity of syntactic forms observable in any language, and thereby show how a uniform system of categories can be applied to all languages. Lastly, I shall discuss how the ideas developed can be applied to the mechanical programming of syntactic analysis.

2. THE CONCEPT OF REPLACEMENT

2.0. ***Replacement in a Closed Language***

We consider a closed language as being a closed corpus consisting of a set of utterances; each utterance ***is*** a sequence of signs having a beginning and an end. The signs in any such sequence are understood to have a unique simple ordering. Each sign may be a written letter or ideograph, or a sound; there are thus various possibilities for the realization of the signs, and in some realizations it may be necessary to resort to special conventions in order that they may be unambiguously assigned a simple ordering; this however, is a matter which at the present level of

discourse need not be pursued in detail.

Any subset of the signs constituting an utterance, presented in the same order in which they occur in this utterance, is called a segment, if S is a segment of an utterance U, and if between the first and the last sign included in S, every sign in U is also a sign in S, then S is said to be an ***uninterrupted*** segment; otherwise, S would be ***interrupted***. We shall have occasion to use the notion of a ***zero*** segment, that is, one consisting of no signs; just as the empty set, in set theory, is understood to be a subset of every set, so we shall admit the presence of an empty sub-segment in every other segment. In all the statements which we shall make about segments, the possibility that a zero segment may be referred to should be borne in mind.

If an interrupted segment consists of n subsegments, each of which is itself uninterrupted, the latter will be called fragments to distinguish them from general subsegments, which may be themselves interrupted. A fragment, being itself a segment, may also on occasion be a zero segment. We shall use, as a general form for denoting a segment, $...F_1 ...F_2 ...$, where F_1 and F_2 are fragments of an interrupted segment. Whenever such a form is used, it must be understood that though two fragments are shown, more than two fragments may in fact be present.

A segment $...F_1 ...F_2 ...$ is said to be ***replaceable*** by another segment $...F_1 ...F_2 ...$ if the following two postulates are fulfilled:

(a) for any X, Y, Z such that XF_1YF_2Z is an utterance in the language, $XF'_1YF'_2Z$ is also an utterance in the language;

(b) for any $...G_1...G_2...$ in the language, of which $... F_1... F_2...$ is a subsegment, there is at least one utterance of the form XF_1YF_2Z in the language, which does not contain $... G_1... G_2...$

The second condition is required to avoid saying that one segment is replaceable by another if they are only so when they are parts of larger ones.

A closed language, as defined above, is a rather unsatisfactory model of actual speech. At the very least it needs to contain an enormous amount of material if it is to provide examples of all possible constructions. Furthermore, in a strict sense, the set of "possible constructions" in any actual language is an open one in that any speaker may coin a new construction without thereby ceasing to speak the given language. We therefore need to pass over from consideration of closed languages, to take account

of open languages.

An ***open language,*** is, like a closed language, considered as a set of utterances. But whereas in a closed language these utterances form an ostensibly given corpus, which can be examined to determine whether a given sequence is or is not an utterance, in an open language the criterion is, whether or not a given sequence is accepted by a competent speaker as a correct utterance in the given language. The definition of replaceability given above, needs modification in three particulars, in order to adapt it for use in an open language. We have to re-define the term "segment"; we have to consider carefully what is implied by a sequence being an utterance; and we have to re-phrase the definition of replaceability.

2.1 ***Segments in an Open Language.***

In effect, we are trying to substitute, for the closed corpus of a closed language, the behavioural response of a competent speaker, to define the compass of an open language. This being so, we cannot simply regard a segment as a sequence of signs, unless we admit as "signs" not only written marks and spoken sounds, but any sensory clue available to the competent speaker during the act of communication. We therefore regard all such clues as imaginary diacritics which could be added to the manifest signs composing a given utterance or segment. In other words, we allow our competent speaker to annotate any text before we subject it to further analysis.

The scope of such annotations may be illustrated by the example of the English phrases 'you and not me' and 'shorthand notes'. Both, as they stand, are sequences of written letters, both can be parts of utterances in English, and both contain the uninterrupted sequence 'and not'. By the definition above, this sequence is certainly a segment, of which both phrases contain exponents. We rely on the annotations or diacritics which a competent speaker might add, to recognise that the two letter-sequences are effectively different. This might, for example, be done by underlining the first and last letters of every word, in which case the two sequences would be 'and not' and 'and not'. The particular device adopted does not matter, provided (a) it can be non-contentiously performed, and (b) it leaves the annotated text capable of complete analysis on the assumption that, if a segment S is replaceable by a segment T, S and T are sufficiently identifiable by the sequences of signs (including the diacritics) which they contain.

If this principle is applied to actual texts in actual languages, it

is possible to find cases where it breaks down. These are cases of irreducible ambiguity. An example is the sentence 'Iceland fish catch drops': it is more than a competent speaker can do to annotate this text so as to distinguish non-contentiously all the meaningful segments in it. For it can bear two distinct meanings, which only a fuller context could disengage: either it concerns animal behaviour, or the fishing idustry, according as 'catch' or 'drops' is taken as the verb. It is therefore necessary to prescind such cases of irreducible ambiguity in the rigorous analysis of open languages.

2.2 ***Recognition of Utterances***

Whereas in a closed language, every sequence of signs either is or is not an utterance, there are four cases which may have to be considered in regard to open languages.
These are exemplified by the following phrases:

```
1.  'It's a nice morning'; This is an utterance in English.
2.  'I'se hungry'        ; Not an utterance; the correct form is
                           'I'm hungry'.
3.  'Lake three stand'   ; Not an utterance, no comments occur.
4.  'Verns hollip'       ; Undecidable.
```

There is no novelty about either (1) or (3). The new cases not paralleled in a closed language are (2) and (4). The last is in fact peculiarly tiresome, in that there are in real life speech situations in which this phrase could be accepted as an utterance, and meaning could be attached to the words 'vern' and 'hollip'. But in the context of any mechanical language processing we have to regard it as not an utterance, because it must always remain unrecognisable, until the words it contains get into the dictionary. The case (2) can be more constructively treated. We shall formulate the following definition:

Def.1.: a sequence S in an open language L which differs from some utterance S' in L, if at all, in such a way that in the given context, a competent speaker of L will unambiguously identify S with S', is said to be ***corrigible*** to S', which is called its ***correction.*** Two different sequences both corrigible to the same utterance are said to be not ***distinct.***

This definition has been so formulated that it applies to the cases (1) and (2) of the above list, but not to (3) and (4). Its effect is, that in open languages, the class of ***corrigible sequences*** will take the place occupied by ***utterances*** in closed languages.

2.3 ***Redefinition of Replaceability***

The definition given for replaceability in a closed language was based on two postulates. The first of these, when its terms are interpreted in the light of what has been said above about segments and utterances, can stand. The second, aimed to exclude recognition of replacement between segments which are "really" parts of larger segments, between which the replaceability relation is more usefully posited, requires amendment. For in an open language it is no longer sufficient, in order to exclude this situation, to find one instance to the contrary, or even a closed set of instances. Thus, in English, we could say that 'ga' is replaceable by 'ra', adducing instances in which 'gain' is replaceable by 'rain'; this is not any the less silly because we can add a few other instances of the same replacement, such as 'gate' being replaceable by 'rate'. Only if there is an open set of such cases can we count the replaceability as genuine.

We are therefore led to the following revised definition:

Def.2. a segment $...F_1...$ $...F_2...$ in an open language L is replaceable by another segment $...G'_1 ...G'_2 ...$ if and only if:

(a) for any X, Y, Z in L such that XF_1YF_2Z is an utterance in L, $XG'_1YG'_2Z$ is a corrigible sequence in L.

(b) for any two distinct utterances XF_1F_2Z the corresponding $XG'_1YG'_2Z$ are also distinct, and

(c) for any segment $...G_1... G_2...$ containing $... F_1... F_2...$ as a proper subsegment, there is an open set of utterances XF_1YF_2Z not containing $... G_1... G_2...$

3. TOTAL PARADIGMS

3.1 ***Equipollence***

As defined above, replaceability is an asymmetrical relation. It can happen that segment S' can replace another segment S while S cannot replace S'. For instance, we can readily show that in English 'them' is replaceable by 'gypsies'. But we cannot replace 'gypsies' by 'them'. For if we make this replacement in the utterance 'the gypsies came', we get 'the them came'. If this is accepted as corrigible, its correction can only be 'they came'. But, 'gypsies came' is also an utterance, dis-

tinct from "the gypsies came". If we make the proposed replacement we get 'them came' which is corrigible, but again corrects to 'they came'. It is not therefore distinct from 'the them came' according to ***Def.1.*** The replacement therefore fails to satisfy postulate (b) of ***Def.2.***

Nevertheless, it is easy to define a symmetrical relation, based on the replacement idea, as follows:

Def.3. two segments S, T in L are said to be *equipollent* if S is replaceable by T, and T by S, in L.

This relationship of equipollence is analogous, at the syntactic level, to that of "replacement" as defined by Jones (10) in regard to semantics. Like the latter, equipollence is a similarity relation; for it is reflexive (every segment is equipollent with itself), symmetrical (by definition), and transitive (for if S is replaceable by T, and T by U, then S is replaceable by U; and conversely). It therefore divides the class of segments in given language into classes, whose members share common syntactical properties, just as Jones' "replacement" divides the class of lexemes into classes whose members share a common "meaning".

3.2 ***Substituents***

However, not all sequences in a given language are either utterances or segments of utterances; likewise, not all segments are recognisable, either by a "competent speaker" or by a trained linguist, as meaningful units of speech. In order to be able to isolate those segments which can be profitably used as units in the syntactic analysis of a text, we need to define a certain subclass of the domain of equipollence which shall contain only those segments which are useful for this purpose.

Def.3. a segment S, interrupted or not, is said to be a ***substituent*** in a language L if there is at least one segment T in L, distinct from S, such that:

(a) T is equipollent with S

(b) there is no sequence U of segments U_1U_2,... such that (b1) for every U_1 there is at least one segment V_j in L distinct from and equipollent with U_1, and (b2) the sequence U is corrigible to T.

The effect of this definition is to recognise as a substituent only segments which are equipollent with ***simple*** substituents, i.e. those which are unable to be further divided into substituents. Roughly speaking,

this allows any meaningful unit, up to a sentence, to be a substituent, since sentences are in general equipollent with single units like "yes" or "no", and in all languages there exist sentences of so formal and stereotyped a character as to be admissible as simple lexemes. For instance, we do not get a true picture of the meaning of 'How do you do?' if we analyze it into its component parts; such a sentence, while certainly equipollent with genuine sentences like "How is your stomach?" is a perfectly good candidate for inclusion as a whole in a dictionary.

It is convenient for some purposes, also, to recognise any sequence of two or more sentences as equipollent with a single sentence; if this is done, the restriction (b) in ***Def. 3 i***s hardly needed. However, we aim eventually to consider the syntactic relations between the sentences in a paragraph or conversation, and for this purpose we must make a fairly clear distinction between "sentences" and higher units which ***Def. 3*** succeeds in doing.

The reason for introducing corrigibility into the postulate (b2) is to allow for words like the French 'au' which while apparently simple substituents (in that they cannot be analysed as they stand into smaller substituents) are inexpedient to admit as such, because in reality they are compounded of units having separate and definable functions in the sentence. But, of course, there exists the sequence 'à le' which, though not a ***segment i***n French, is certainly ***corrigible*** to 'au', and which is a sequence of segments each equipollent with at least one other ('à with 'dans'; 'le' with 'un'). The reason why we do not want to have to treat 'au' as a single substituent, is that in an expression such as 'au fond' we would like to recognise as substituents not 'au' and 'fond' but the more logical pair 'à' and 'le fond'. In bracket notation we would wish to analyse 'au' into 'à (le...)'.

The following supplementary definition therefore suggests itself for use in connection with substituents: ***Def. 4.*** a substituent of S in L is said to be ***compound*** if it is the correction of * a sequence U of segments U_1U_2,... such that

(a) each U_1 is a substituent in L, and

(b) the sequence left on replacing any one U_i by the zero segment is also a substituent in L.

---

* note here, that by ***Def.1*** every ***segment*** is its own correction.

In such a case, the segments U_1, U_2,... are the ***components*** of S.

3.3. ***Paradigms***

We have already mentioned that equipollence is a similarity relation dividing any subclass of its domain, and in particular the class of substituents, into equivalence classes. Members of any of these classes would be said, by linguists, to have the same syntactic function. However, the following definition proves to be more amenable to our purposes:

Def. 5 the ***total paradigm*** of a substituent S in a language L is the set of all substituents in L which contain either S, or another substituent equipollent with S, as subsegments.

It is part of the method of this work to replace the unsatisfactory unit of the "word", already abandoned by most linguistic schools, by the carefully defined concept of "substituent". It is this replacement which justifies the use of the term "paradigm" in this sense. It will shortly appear, that those members of the total paradigm of a "stem" (which, in an inflected language, is in general a simple substituent) which are "words" in the conventional sense form a set almost Identical with the "paradigm" in the traditional linguists' sense.

It is evident that if two substituents S, T are equipollent, then according to Def. 5 they must belong to the same total paradigm. Moreover, if T is not equipollent with S, then either (a) T contains S as a proper sub-segment; in which case S, which is contained in the paradigm of S, is not in the paradigm of T; or (b) S contains T, with the complementary effect; or (c) neither S nor T contains the other, in which case both paradigms contain substituents not in the other. Therefore, if S, T are not equipollent, they belong to different total paradigms. Thus, the total paradigms defined in ***Def. 5*** are indeed equivalence classes under equipollence.

The relation between total paradigms, and the syntactic functions of the linguist, is now clear. If any two substituents belong to the same paradigm, then they share a common function. If they belong to different paradigms, they have no common function, unless their paradigms have a non-trivial union, in which case the latter provides them with a common function. We therefore postulate a one-to-one correspondence between syntactic functions and total paradigms; the first are the properties which characterize the second as classes.

However, in formal statements I shall prefer the term paradigm to function, on the grounds that the latter word has too many other uses not entirely excluded by the context. I shall normally drop the epithet "total" before "paradigm", where no confusion is likely to follow.

Thus, while we have this simple relationship between our total paradigms and the relation of equipollence, their structure under the relation of replaceability is somewhat more complex. It may be reduced to the following five Lemmas:

1. If a substituent A replaces both B and C, where B, C are not equipollent with each other, then the paradigm of A is the set union of those of B, C.

2. If a substituent A consists of two or more segments, each a substituent, B, C...., the paradigm of A is included in each of those of B, C,...

3. If there is in a language L a substituent Z such that any other substituent Z' containing Z is equipollent with Z, then the paradigm of Z is contained in that of every substituent.

4. If there is in L a segment which can replace every other substituent in L, then this segment is a substituent in L, and has a paradigm including those of all other substituents in L.

5. The paradigm of any substituent is unique (provided we take due account of the procedures mentioned in Sec. 2.1).

The substituent Z mentioned in Lemma 3 is exemplified by a complete sentence not forming part of any other sentence and associated only by concatenation with other segments in an utterance. Formally we may state the following:

Def. 6. A substituent in a language L is a ***free sentence*** of L if it is a component of an utterance equipollent with the whole utterance.

The segment mentioned in Lemma 4 is exemplified by a sign of omission such as ..., or a word such as 'thingummy' used to replace any word which a speaker will not trouble accurately to recall.

The above five lemmas are sufficient to prove that, if we assume the existence of the substituents postulated in (3) and (4), the system of

all the total paradigms is a ***lattice*** under the set-inclusion relation. For if S, T are any two non-equipollent substituents, their respective paradigms have, potentially, a ***join*** defined by (1) and a ***meet*** defined by (2), while the bounds of the lattice are provided by (3) and (4); these points satisfy the definition of a lattice (see Birkhoff (1))

4. CONSTRUCTION OF THE GENERAL SYNTAX LATTICE

4.0 *The Primary Syntax Lattice*

Having established that the system of paradigms of substituents in any language must form a lattice, we have now to show what lattice is in fact formed. This could be done empirically, by applying the definitions given above to a sufficient body of texts in a given language. Even with the best mechanical aids, this would be a virtually impossible task, even for one language. Or, it could be done intuitively; any intelligent person can learn how to do this for a language he knows well enough, but the results carry conviction only to one who has himself gone through the procedure. I shall therefore construct the syntax lattice, step by step, starting from the free clause and introducing progressively finer syntactic contrasts, till it is sufficiently developed to serve as a model of actual language; I shall then show how it can be used in the design of a syntax analysis program.

We have already seen that, in a lattice representing the total paradigms of substituents, if a point A 'includes' (or 'precedes') a point B, a substituent whose paradigm is A can form part of a substituent whose paradigm is B. We can assume that the simplest imaginable 'language' has the capacity for some kind of syntactic contrast, and has sentences made up of smaller units. The syntax lattice for such a rudimentary language would therefore be as shown in ***figure 1.***

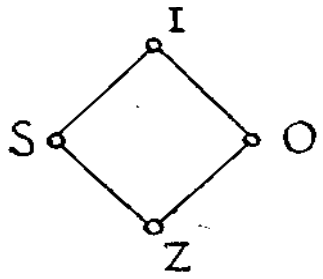


Figure 1. The simplest stage.

Each point in this lattice represents the total paradigm of some substituent; that is, the syntactic function of a class of substituents in the model language. We name these as follows:

S = substantive function
O = operative function
I = indeterminate function (for substituents which can be used either as substantive or as operative).
Z = function of compound substituent made up of at least one substantive and one operative component.

Even in this simple schema, the following lattice properties are illustrated:

(1) the side-to-side symmetry of the lattice: i.e. the complementarity of O and S. This distinction we interpret as the subject - predicate dichotomy.

(11) the top-to-bottom asymmetry of the lattice: i.e. the partial-ordering relation (inclusion of paradigms defined as sets). At this stage, this has only a trivial interpretation, but later we shall correlate this with the governor - dependent distinction.

(111) the two binary operations definable in every lattice, namely the ***join*** and ***meet*** of any two points, denoted by $\cup$ and $\cap$ respectively. As we have already seen, we are committed to interpret $a \cup b$ as the syntactic function of a substituent capable of being used either like those of function a or like those of function b; and $a \cap b$ as the function of a substituent having components of functions a and b.

The lattice just considered, with its two principal points O, S, gives us our basic schema. But it is clear that it is far too primitive as it stands for its use to be extended from our imaginary language to any real one. To obtain a more adequate system, the basic schema is therefore enlarged by the addition of points representing new paradigms, which include, in their capacity as sets, the points S and O respectively.

This gives us the lattice shown in ***figure 2.***

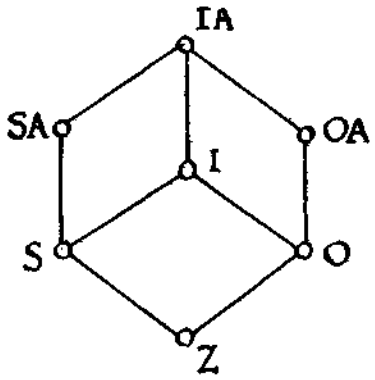


Figure 2. The next stage

It is obvious that if we are to retain the substantive-operative distinction represented by the two sides of the lattice, we cannot extend the system in any other way, for the more refined classification we require must represent a sophistication of this basic division. The two new side points SA and OA represent substantive adjuncts and operative adjuncts respectively. For if ***book*** has the paradigm S and ***new book is*** equipollent with ***book,*** the paradigm of ***new*** includes both ***book*** and ***new book,*** whereas that of ***book*** does not include ***new;*** such examples, in the light of ***Def. 5,*** show that SA and OA stand, very roughly, for adjectives and adverbs. Their join gives us the new indeterminate adjunct IA, and this also includes the principle indeterminate I. The lattice now has seven points.

This extended lattice is still however, inadequate, and we need to add, above SA, OA, IA a further series of paradigms SB, OB, IB to represent subadjuncts, that is, for words whose use is restricted to qualifying other adjuncts. This gives us a ten-point lattice; but this still fails to account for certain syntactically important types of words, such as prepositions, conjunctions, etc. Prepositions could, without too much arbitrariness, be classified as postverbs, and assigned to the point OB, but conjunctions (connectives, as the logician understands them) are still unaccounted for. Since the join operation is the lattice equivalent of the logical ***and/or*** connective, we can expect to represent conjunctions by an additional point at the very top of the lattice, IC. Indeed, there are conjunctions in some languages which can be used to connect words of any syntactic function, and whose paradigm therefore by ***Def. 5*** includes all other paradigms. But this does not go for all "conjunctions". There are those which specifically connect clauses rather than separate words; these have a paradigm ZA immediately including Z. The lattice now has

12 points, and has the graph shown in ***figure 3.***

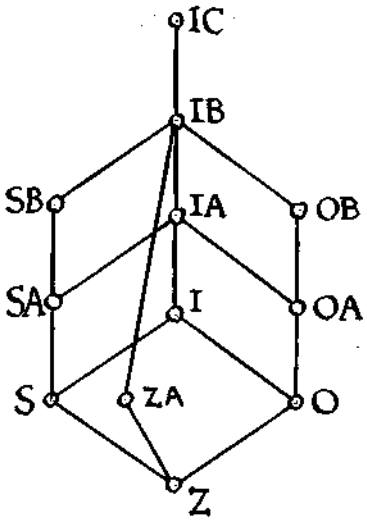


Figure 3. Complete Primary Lattice

Preliminary empirical investigation has shown that, to a surprising extent, this system contains the hard core of the general syntactic classification we need. It is therefore called the ***primary syntax lattice;*** that point of the lattice representing the paradigm (i.e. syntactic function) of a substituent, is called the lattice position indicator or LPI of the substituent.

4.1. *The Meet Algorithm*

Having set up the classification represented by the primary lattice, we can now start to use it to find the function of a compound substituent. I here introduce the first algorithm derived from the theory, which I call the "meet algorithm". This is simply Lemma 2, in the form: The paradigm of a compound substituent is the meet of those of its components.

Let us see how this algorithm works out in the lattice of ***fig. 3,*** The meet of the points SA and S, for instance, is S. That is to say, a group consisting of a substantive and a substantive adjunct has the function of a substantive (a man with long legs: equipollent with a man); similarly, the meet of O and OA is O (you have finished it? I have!).

For either side of the lattice, therefore, the algorithm works in a satisfactory way and gives linguistically acceptable results; clearly, if we put together units with a common character, we should expect the group to have the same character.

The meet algorithm falls, however, when we consider the meet of any two points on opposite sides of the lattice. For such a meet can only be the point Z, whereas in fact a great variety of syntactic functions can be discharged by substituents having components between which the substantive-operative contrast is in evidence. Given this complementarity, we can only describe Z, very weakly, as the property of having a function different from those of its components. This is serious, since we began by interpreting Z much more strongly, as the paradigm of a free clause.

4.2 *Endocentric and Exocentric Substituents*

In order to deal with this situation, we make use of the distinction between an ***endocentric*** substituent, which is equipollent with one or more of its components; and an ***exocentric*** substituent, which is not equipollent with either. What we have found is that the meet algorithm, applied to the simple primary lattice of ***fig. 3***, works for endocentric but fails for exocentric substituents. We must now develop the lattice schema to take account of exocentric substituents; in fact, the strength of the theory largely rests in its capacity to give an adequate and precise account of the nature of exocentric substituents.

The key to this development is to make use of the lattice relation of duality. Just as all the other points in the primary lattice represent possible parts of what we at first interpreted as a clause, represented by the lower bound Z: so, now that we have to weaken the interpretation of Z to that of an exocentric substituent, we need lattice points to represent all those substituents of which an exocentric group Z is a possible part. Of this new set of points, by lemma 2, Z is thus the ***upper*** bound. And since we may expect a substituent of ***any*** function to have components replaceable by exocentric groups, the new set of points must contain all those, besides Z, which we have already allowed for in the primary lattice. What we have to do, therefore, is to add to the primary lattice its own ***dual*** (consisting of the same points with the converse inclusion-relation between them), the point Z being in common between them.

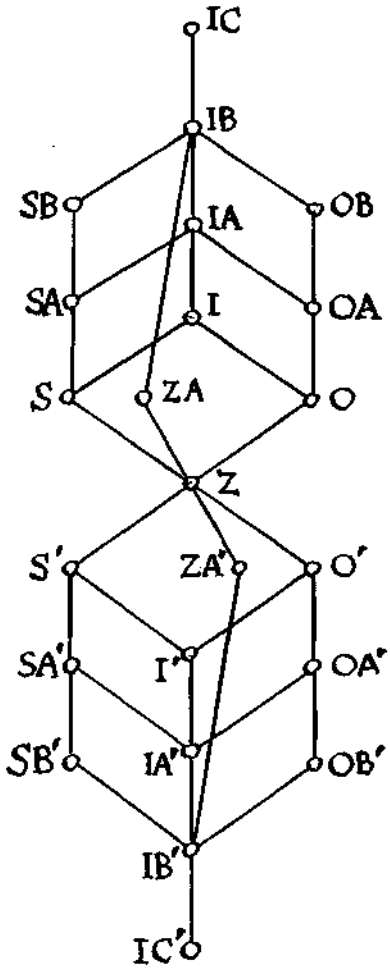


Figure 4.

The Primary and Secondary Lattices Combined.

This system, shown in ***figure 4***, is still a lattice; it is divided up into two mutually dual sublattices, the primary lattice which, as we have seen, is concerned with endocentric substituents, and the secondary lattice, as we may now call it, which is concerned with exocentric substituent.

This lattice however is still not the one we want; the meet algorithm applied to points in the primary sublattice still gives Z as the result for all exocentric groups. To avoid this, we have to accept the existence of further distinctions. There is not, for instance, just one kind of

operative, O: there must be different kinds, each determining a different function for the exocentric groups which it can enter into. Thus, to the different kinds of exocentric groups represented by the dual secondary lattice hanging from Z, there must be added a parallel set of distinctions hanging from every other point of the primary lattice, representing the different kinds of operatives, of operative adjuncts, and ***so*** on.

The resulting system, when fully developed, as will be clear to those acquainted with lattice theory, will be the direct product of the primary lattice of ***fig. 3*** with its dual. This lattice is too large for convenient setting out here. But we can take advantage of the fact that in actual practice we can do with a less complete classification of functions for compound substituents than is required for their irreducible components. Thus, so long as we are only interested in compound substituents, we can replace the 12-point primary lattice by the 5-point lattice shown in ***figure 5***.

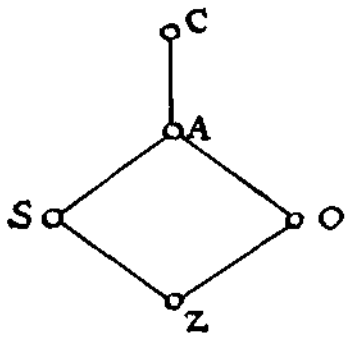


Figure 5. Simplified Primary Lattice

Note: it is ***not*** claimed that this lattice is actually adequate for all languages, even in the sense that the lattice of ***figure 3*** is adequate; it is used here because (a) it provides a sufficient basis for the classification of ***compound*** substituents and (b) it gives a self-dual-product lattice compact enough for convenient illustration. This product lattice is shown in ***figure 6***.

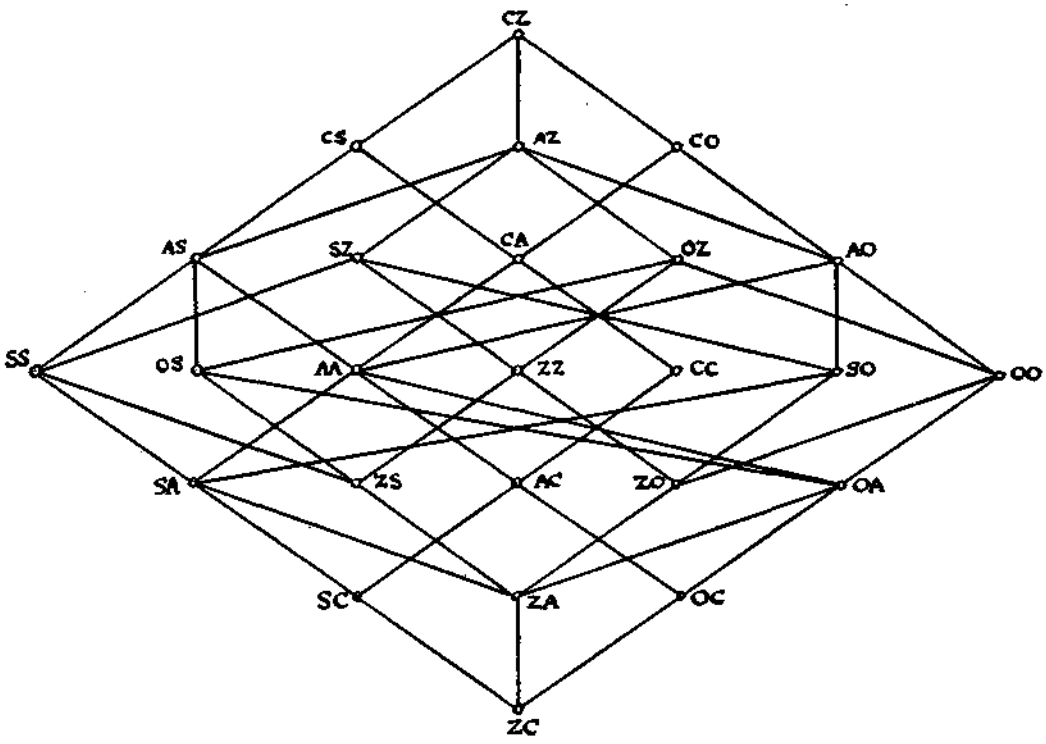


Figure 6

Simplified Self-dual-product lattice

This, by the way, is the smallest possible self-dual-product lattice.

4.3 *The Polar Algorithm*

We can now see how the meet algorithm works out in this fuller version of the syntax lattice. For endocentric groups there is no problem; for by definition one of the points concerned must include the other, which, as before, is their meet and defines the function of the whole group. But for exocentric groups the situation is more complicated.

Some exocentric substituents will have a meet in the lower ideal of ZZ, that is, in the lowest exponent of the secondary lattice. Now we have already indicated that to these points we attach the same syntactic functions as are attached to the corresponding points in the primary lattice. However, if we make no modification in the meet algorithm, we shall be faced

with the difficulty that if any such substituent is in turn included as part of a yet larger one, the function of the latter can be represented only by a point yet lower in the lattice; in fact, if an exocentric group forms part of another exocentric group, the latter will be assigned the function Z.IC, which is that of a conjunction and is unlikely to be correct. To cope with this situation we make use once again of the top-to-bottom contrast in the lattice. What we need is to get from a point in the lower ideal of ZZ to a point in its upper ideal. Because of the duality relation between these two sublattices, their respective points are already in correspondence; in the notation used in ***figure 6***, those in the upper ideal have letter-pairs ending in Z, and those in the lower ideal have letter-pairs beginning with Z. Therefore, we make the rule that whenever the meet algorithm leads us to a point in the lower ideal of ZZ, we replace this, as the result of the algorithm, by the corresponding point in the upper ideal of ZZ. This rule I call the ***polar algorithm.***

On this basis, exocentric groups can be divided into three classes, all of which share the property that their functions differ from those of any of their components. In the first class, the function of the whole group is ZZ; in the second, it is a point in the lower ideal of ZZ; in the third, it is any other point in the lattice. Linguistically, these three classes represent different degrees of "completeness"; the first are complete clauses, the second may be called subordinate clauses, while the third class contains groups too incomplete to be called clauses at all.

6. BASIC LANGUAGE CONTRASTS IN TERMS OF THE THEORY

5.1 *The Governor - Dependent Relation*

In the system, represented in a simplified form in ***figure 6***, we can interpret certain of the lattice relations in more detail than has been shown above. One additional descriptive contrast which the theory thus gives us is that between the governor and the dependent of any compound substituent. One finds, in any such group, that there is one component which 'colours' or 'gives tone to' the whole group, the others having a more passive role. Thus, in any substituent with three or more components, one will stand out from the others; this we call the ***governor***, and the other components are the ***dependents*** of the substituent

(Note: dependents are "of" substituents, not "of" the associated governors). If all available examples of a given substituent type have only two components, we must identify the governor from its lattice properties.

Now in terms of the lattice, this works out differently in the three cases of exocentric groups, endocentric groups, and conjunct groups. In a free clause, it is clearly the verb-group which gives its colour to the whole; thus we make O governor over S. The same rule will serve for all exocentric groups, in which the primary functions of the components show this difference; where they do not (as for instance in a group whose components are SZ and AA) we may go by their secondary functions (in the case mentioned, as these are respectively Z and A, this means treating it as if it were an endocentric group). In the complete 144-point lattice, difficulty may also arise from components with I functions: in any particular context, these may behave in their S or in their O capacity, and this must be ascertained first.

In endocentric groups, it is clearly the component which has the same function as the whole which colours the group; thus, the component represented by the lowest point on the lattice is the governor. In this case, then, the dependent-governor relation is straightforwardly the inclusion relation in the lattice. Clearly, too, while we can have such a group with several upper points, the meet algorithm forbids the existence of an endocentric substituent with more than one lower bound, namely, the governor.

In conjunct groups, the dominant component is the conjunction itself; though, standing as it does at the top of the lattice, it does not affect the group's function. Thus, we adopt the convention that, when any point which is a join in the lattice is interpreted as the join-relation itself, this shall mark the governor of the group.

5.2 ***The Subject - Predicate Relation***

A yet more important place in traditional linguistic description belongs to the subject-predicate contrast than to the governor-dependent contrast, and this too can be derived very simply from the present theory. We have seen that, by means of the polar algorithm, a scale of increasing completeness of exocentric groups can be defined. Incomplete clauses have their meets outside the lower ideal on the point Z.Z; complete but subordinate clauses have meets within this ideal, which the polar algorithm transposes into the upper ideal; a free clause has its meet actually at

Z.Z, which the polar algorithm transforms into itself, and which thus represents a point of stability or finality.

Substituents of this last type, and these alone, are subject - predicate groups. This theory of syntax therefore provides a representation for the subject - predicate pattern in language: it is that of an exocentric substituent whose meet is at the point Z.Z, one of the two non-bounding vertices, or "central" points, of the lattice. It can also be thought of as the result of applying the polar algorithm as a stop rule: when in the build-up of a sentence structure we come to this point we can stop, but not before.

This interpretation does not at first sight seem to have much to do with the logicians' notion of subject and predicate. These have been thought of either grammatically (as "sentence with a main verb") or, following Russell, formally (as a formula of the type xP subject to extension either by the addition of further terms y, z, ... to the x, or by adding quantifying restrictions to the x). As to the grammatical interpretation, our theory explains rather the notion of mainness than of verbness; it shows us how to build up the distinction between "main" and "subordinate" verbs (all verbs being initially merely operatives). As to the logical interpretation, the theory does not define the relation of predication (though it can cope with the distinction between monadic, dyadic etc, relations), but it does explain why, however predicative logic is developed, the symbol P remains unique and unchanged; for this point, our Z.Z, is one of the four vertices of the lattice which can be transformed, by inversion of factors in the lattice, into the upper or the lower bound, just as in logic P is the point from which, no matter how far the x - sequence is extended the whole system of relation always hangs. It is in this sense that it is possible to say that, starting from acknowledged linguistic notions, which we define more exactly than heretofore, we can arrive at an account of this important logical form, the subject - predicate sentence.

7. THE APPLICATION OF THE MODEL

6.1 ***Substituent types and participation classes***

We must now show how the coding used in the empirical procedures currently being tested on the Cambridge computer is derived from the theory. The actual program is fully discussed elsewhere (11), but it

will be illustrated here by a simplified example.

The full product lattice has 144 elements, and in principle substituents with any pair of these functions may form a group, or compound substituent. As the model does not take the order of the components of a group into account we thus have ½ (144^2) possible different groups. The function of a group is, however, determined by the meet algorithm, and the group formed by any one of the 10368 possible pairs of substituents must therefore be defined by a point in the lattice. There are thus only 144 different meet-points. 12 of these points, moreover, are in the lower ideal of zz, and are not accepted as they stand but converted by the polar algorithm. This leaves us, therefore, with 132 result-points representing different kinds of group; these will be called ***substituent types***.

By using this set of substituent types we can extend our classification system. In setting up the syntax lattice we treated it as a schema for classifying substituents according to their functions. The function of a substituent is naturally related to its behaviour in groups of substituents, but we did not initially attempt to classify substituents according to the kinds of group in which they can figure. It will now be clear that we can give more information about a substituent if we take what we may describe as its grouping possibilities into account. These are derived from the lattice in a straightforward way: for a substituent with a particular function we list the set of result-points which can be reached when operating on a pair of substituents, one of which is the substituent in question. We thus give, in terms of their functions, the kinds of group in which the substituent can participate. This information is represented by a positive mark in the appropriate positions in a 132-place entry. We will call the entry as a whole the substituent's ***participation class***. The record is further refined by entering, for each kind of group in which the substituent can figure, whether it functions as governor, or dependent, or either.

For practical purposes, however, the size of participation class entries as just described is not very satisfactory: thus it is too large for convenient machine handling, or for teaching to dictionary makers. It can, however, be reduced as follows. In terms of the theory the set of 132 result-points can be naturally divided into those which lie in the principal exponent of the primary lattice, and those which fall elsewhere, that is, into those with secondary function Z. (Except for ZZ, points with primary function Z are excluded by the polar algorithm.) It will be clear from the lattice that there are 12 points of the first kind and

120 of the second. This distinction represents the extent to which further grouping is required before the stop-point defined by ZZ can be reached. Compound substituents with secondary function Z can be 'direct' components of full clauses; those with neither function Z require at least one intermediate grouping, with the application of the polar algorithm, before they can be grouped to give a full clause. It can be argued that the information about a substituent represented by the fact that it can be a member of a group of the second kind is less useful that that representing its membership of a group of the first kind, given that from a group of the second kind one of the first kind will be reached. If we accept this argument, we can then replace the 132-place participation class by one with 12 places. For a particular substituent this replacement will give the result-points in the principal exponent of the primary lattice which will be reached by operations on pairs of substituents of which the substituent in question is a member.

Examination of natural languages shows that the 12-place participation class is an oversimplification. Thus in English there are two kinds of compound substituent which would both be given the function OA.Z, namely adverbial groups (like "almost exactly") and adverbial clauses (like "considering the circumstances"). These clearly represent different constructions and if they were identified in classifying the behaviour of a word, would lead to incorrect grouping. We can, however, deal with this difficulty by taking into account distinctions which the theory already contains. For instance, we can at least make use of the distinction between substituents, the meet of whose component functions falls in the lower ideal of Z.Z, and those whose meet falls elsewhere in the lattice; that is, be-, tween complete exocentric groups (clauses) and the rest. This division would take us from 12 functions to 24 substituent types, straightforwardly derived from the theory, and therefore of interlingual validity (though we must not expect that all of them will be represented in any particular language.) In particular, we can construct model and restricted languages requiring many less distinctions than this. Most natural languages appear to need between 10 and 20 substituent types for their adequate analysis,

6.2 ***Bracketing***

We have so far discussed grouping, or bracketing, in terms of lattice points and lattice algorithms. We must now show how this works out for actual texts. Given that each substituent type defines a kind of group, it is clear that the fact that a set of substituents can be bracketed will be represented in their respective participation classes by a positive

entry for the same substituent type. This in itself, however, is not enough; the items to be bracketed must also be contiguous, and must satisfy the governor dependent relation. The latter means that we can only bracket a group of substituents if one of them can be the governor, and the rest dependents, in the kind of group concerned. The governor-dependent relation thus acts as a restriction on bracketing.)

Two points should be noted: (1) AS many items as possible can be combined at the same time to form a group. (2) The substituent types are arranged in a priority order from left to right: that is, we look for groups of kind 1 first. The order loosely corresponds to the lattice structure in that 'weak' groups are found first, and full clauses last, but it is essentially a practical device for reducing the amount of effort spent in trying to find brackets: as bracketing is carried out on ever larger units, there is clearly some point in looking for the smallest groups of most closely associated substituents first.

The way in which the information contained in participation classes is used for bracketing can be illustrated as follows:

	A	B
x	+	-
y	+	+

This means that x can belong to a group of kind A, but not of kind B, and that y can belong to groups of kind A and kind B. If x and y occur in contiguous positions in a text and can therefore be bracketed, the resulting group must have the function A.

We will now consider a more elaborate case with governor-dependent information given:

	A	B	C
x	G	-	D
y	D	D	-
z	-	-	D

When the governor-dependent rule is satisfied only x and y can be bracketed,

to give a group with function A. x and z are both dependents in groups of type C and cannot therefore be combined.

6.3 ***Simplified example***

In order to keep the example simple the following modifications of the actual procedure have been made:

(i) habitat and concord information is omitted;
(ii) only 6 substituent types are used;
(iii) the bracketing rules are formulated rather crudely.

The sentence to be bracketed is:

A rather lazy cat chases falling leaves and butterflies; of course these can easily get away.

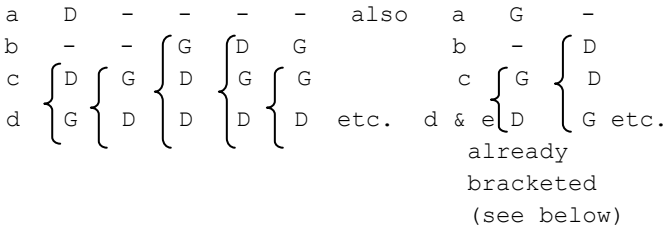
We will assume that the participation class entry for each word has been obtained by dictionary look-up. The sentence with appropriate entries is as follows:

		1	2	3	4	5	6
		C	SA	S	OA	O	Z
a	SA	-	-	D	-	-	-
rather	SB		D				
lazy	SA	D	G	D	-	D	-
cat	S	D	-	G	-	-	D
chases	O	D	-	-	-	G	G
falling	IA	D	D	B	G	D	G
leaves	I	D	-	G	-	G	B
and	C	G	-	-	-	-	-
butterflies;	S	D	-	G	-	-	D
of course	ZA	-	-	-	-	-	D
these	S	D	-	-	-	-	D
can	O	D	-	-	-	G	G
easily	IB	D	D	-	D	-	-
get	OA	D	-	-	G	D	G
away	OB	D	-	-	D	-	-

Bracketing is carried out according to the following rules:

Rule 1

Starting at the last item before the punctuation stop, whether simple or a compound obtained by previous bracketing (see below), read backwards in each column in turn, looking for the longest continuous sequence immediately preceding the stop in which one item is governor and the rest dependents. As the priority rating of the columns is from left to right the first such sequence is taken (even if there is a longer one in a later column). For example, in a particular column the following are all bracket groups:



No brackets starting from c can be obtained in the following:

A	-	D	G
b	D	G	-
c	D	-	D

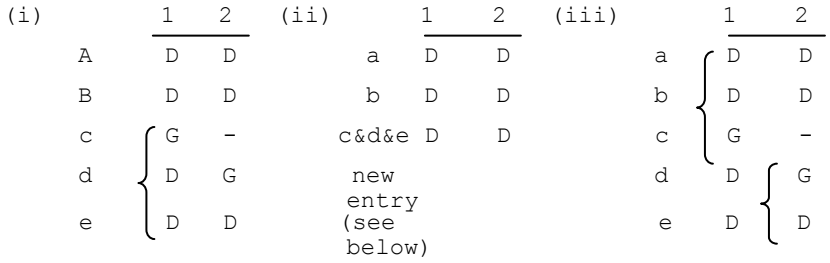
Rule 2

When ***Rule 1*** suggests a bracket in column 1 if the item marked as governor (i.e. the conjunct substituent) is immediately flanked by two items marked as dependent, treat the three as a group. Thus the first case below will bracket but the second will not:

a	D	D
b	D	D
c	G	D
d	D	G

Rule 3

If under ***Rule 1*** in proceeding backwards from a group already made no brackets can be found, take from the beginning of the existing group the smallest number of items compatible with its remaining a group and try backwards from the last of these before trying again with the reduced following group. There is shown in the following example:



6.4 ***Treatment of bracket groups obtained***

The column in which a bracket is made represents the type of the resulting compound substituent. Reference to ***Table I*** below gives the appropriate participation class entry, and the group with this new entry is treated as a single item in further bracketing.

TABLE I

Participation class entries for compound substituents of each type:

		1	2	3	4	5	6
		C	SA	S	OA	O	Z
2:	SA	D	-	D	-	D	-
3:	S	D	-	G	-	-	D
4:	OA	D	-	-	-	D	G
5:	O	D	-	-	-	G	G
6:	Z	D	-	-	-	-	G

The participation class entry for a compound containing 1 : C is the meet

of the entries for the dependent items.

Bracketing can now be carried out as follows:

Stage 1

By ***Rule 1*** we can bracket in column 4 the last three items:

a	-	-	D	-	-	-
rather	-	D	-	-	-	-
lazy	D	G	D	-	D	-
cat	D	-	G	-	-	D
chases	D	-	-	-	G	G
falling	O	D	B	G	D	G
leaves	D	-	G	-	G	B
and	G	-	-	-	-	-
butterflies;	D	-	G	-	-	D
of course	-	-	-	-	-	D
these	D	-	-	-	-	D
can	D	-	-	-	G	G
easily	D	D	-	D	-	-
get	D	-	-	G	D	G
away	D	-	-	D	-	-

By referring to ***Table I*** we give this group the participation class entry for a substituent of the type represented by the column in which the bracketing was made, i.e. 4:

D - - - D G

Stage 2

By ***Rule 1*** we can bracket in column 5 the group just made and the preceding item:

a	-	-	D	-	-	-
rather	-	D	-	-	-	-
lazy	D	G	D	-	D	-

cat	D	-	O	-	-	D
chases	D	-	-	-	D	O
falling	D	D	B	G	D	G
leaves	D	-	G	-	G	B
and	G	-	-	-	-	-
butterflies;	D	-	G	-	-	D
of course	-	-	-	-	-	D
these	D	-	-	-	-	D
can	D	-	-	-	G	G
easily get away	D	-	-	-	D	G

We give this group the participation class entry:

D	-	-	-	G	G

Stage 3

By ***Rule 1*** we can bracket in column 6 the group just made and all the preceding items back to the stop-point marked by the semi-colon

a	-	-	D	-	-	-
rather	-	D	-	-	-	-
lazy	D	G	D	-	D	-
cat	D	-	G	-	-	D
chases	D	-	-	-	G	G
falling	D	D	B	G	D	G
leaves	D	-	G	-	G	B
and	G	-	-	-	-	-
butterflies;	D	-	G	-	-	D
of course	-	-	-	-	-	D
these	D	-	-	-	-	D
can easily get away	D	-	-	-	G	G

This is correct as the group formed is of the type complete clause.

Stage 4

Re-starting from the semi-colon, by ***Rule 2*** we can bracket the last three items:

a	-	-	D	-	-	-
rather	-	D	-	-	-	-
lazy	D	G	D	-	D	-
cat	D	-	O	-	-	D
chases	D	-	-	-	G	G
falling	D	D	B	G	D	G
leaves	D	-	G	-	G	B
and	G	-	-	-	-	-
butterflies	D	-	G	-	-	D

We give this group the participation class entry:

D	-	G	-	-	D

Stage 5

By ***Rule 1*** we can bracket in column 3 the group just made and the preceding item:

A	-	-	D	-	-	-
Rather	-	D	-	-	-	-
lazy	D	G	D	-	D	-
cat	D	-	G	-	-	D
chases	D	-	-	-	G	G
falling	D	D	B	G	D	G
leaves and butterflies	D	-	G	-	-	D

We give this group the participation class entry:

D	-	G	-	-	D

Stage 6

By ***Rule 1*** we can bracket in column 6 the group Just made and the two preceding items:

a	-	-	D	-	-	-
rather	-	D	-	-	-	-
lazy	D	G	D	-	D	-
cat	D	-	G	-	-	D
chases	D	-	-	-	G	G
falling leaves and butter-flies;	D	-	G	-	-	D

We give this group the participation class entry:

D - - - - G

Stage 7

By ***Rule 1*** this group will not bracket with preceding Items:

a	-	-	D	-	-	-
rather	-	D	-	-	-	-
lazy	D	G	D	-	D	-
cat chases falling leaves and butter-flies	D	-	-	-	-	G

By ***Rule 3*** the first item in this group will bracket in column 3 with the preceding item:

a	-	-	D	-	-	-
rather	-	D	-	-	-	-
lazy	D	G	D	-	D	-
cat	D	-	G	-	-	D
chases	D	-	-	-	G	G
falling leaves and butterflies	D	-	G	-	-	D

We give this group the participation class entry:

D - G - - D

Stage 8

By ***Rule l*** this group will not bracket with preceding Items:

a	- - D - - -
rather	- D - - - -
lazy cat	D - G - - D
chases falling leaves and butterflies;	D - - - - G

By ***Rule 3*** the first item in this group will bracket in column 2 with the preceding item:

a	- - D - - -
rather	- D - - - -
lazy	D G D - D -
cat	D - G - - D
chases falling leaves and butterflies;	D - - - - G

We give this group the participation class entry:

D - D - D -

Stage 9

By ***Rule 1*** this group will not bracket with the preceding item:

a	- - D - - -
rather lazy	D - D - D -
cat chases falling leaves and butterflies;	D - - - - G

By ***Rule 3*** the first item in the following group will bracket in column

3 with this group and the preceding item:

a	-	-	D	-	-	-
rather lazy	D	-	D	-	D	-
cat	D	-	G	-	-	D
chases falling leaves and butterflies	D	-	-	-	-	G

We give this group the participation class entry:

D - G - - D

By ***Rule l*** we can bracket in column 6 the two groups:

a rather lazy cat	D	-	G	-	-	D
chases falling leaves and butterflies	D	-	-	-	-	G

This is correct as the group formed is of the type complete clause.

We now have the whole sentence bracketed as follows:

((a(rather lazy) cat) (chases (falling (leaves and butterflies;))))
(of course these (can (easily get away.)))

6.5 *Application to English as a Particular Language*

When we come to apply the methods described to a particular language, the following procedures have to be gone through. So far, these have been effected only for English, though plans are now made to apply the same methods to several other languages.

First, we have to fix upon a suitable set of substituent types to describe the constructions met with in the given language; in making this choice, considerations of informational efficiency will play a large part; for in most languages it is possible to find a few examples of substituent types which it is not practically expedient to recognise because of their rarity or difficulty of recognition. Next, given our substituent types, we have to prepare participation classes based on them, to act as the syntactic parts of the dictionary entries for each word in our dictionaries.

Then we have to draw up programs to cover the unilingual operations required to bring the given language into a state analysable by the interlingual part of the program.

A program embodying these procedures for English is being tested on the Cambridge University computer EDSAC II. Up to date its performance has been satisfactory, though large scale testing has not yet been begun.

REFERENCE

BIRKHOFF, G. Lattice Theory (2nd. edn.) ***Amer. Math. Colloq. Publications***. 1954.

CHOMSKY, N. Syntactic Transformations. The Hague, 1957.

ROBINS, R.H. In Defence of W.P. ***Trans. Philol. Soc***. 1959, p.112.

HARRIS, Z. Methods in Structural Linguistics. Chicago. 1955.

HIZ, H.I. The Intuitions of Grammatical Categories. ***Univ. Penna***., ***Transformations and Discourse Analysis Projects***, 1960. No.29.

HOCKETT, C.F. Two Models of Grammatical Description. ***Word***, 1954. **10**, p.210

HUSSERL, E. Logische Untersuchungen (2nd. ed.) Halle. 1915.

KAUFMAN, B. Iterative Computation of String Nesting. ***Univ. Penna***., ***Transformations and Discourse Analysis Projects***, 1960. No.20.

SAPIR, E. Language. New York. 1921.

JONES, K.S. Mechanical Semantic Classification. This conference

PARKER-RHODES, A.F. et al, (1961). The CLRU Computer Program for Syntactic Analysis CLRU, ***ML-136***.