TOWARDS A COMPUTER-TESTABLE WORD GRAMMAR OF ENGLISH

Richard HUDSON

Abstract

The paper summarises the work which I did during a research year (1987-8). My aim was to develop a computer system, in Prolog, which would allow me to write grammars and test them against sample sentences. The system would assume the Word Grammar theory of grammatical structure, and would be applied to a Word Grammar for a fragment of English. The outcome of the work was a working system, consisting of a parser and a grammar-fragment. The parser works reasonably efficiently and is sufficiently general to be combined with Word Grammars other than the one in the system. It processes sentences incrementally, yielding as many readings for each word as are compatible with the words processed so far. The grammar-fragment deals with some quite complex phenomena, including gapping, but it is still very restricted in coverage.

1. Aims

The aim of the research was to produce a computer system which would combine a Word Grammar (WG) grammar (Hudson 1984, etc.) for a fragment of English with a parser, so that the grammar could be tested against selected sentences. The sentences were to be written in conventional orthography, with word-spaces, so the work was clearly in the area of language-processing rather than speech-processing.

The aim seemed realistic, given that I had already developed a small parser for a WG grammar, and a more sophisticated parser seemed to need only time and computer support; and other, larger-scale, parsers for WG grammars had also been developed by other people (Norman Fraser and Max Volino), while others were actively working on them (Barbara Gorayska and Phil Grantham); see Fraser 1985, 1987 and Grantham 1987 for documentation of these early efforts. As it turned out I had seriously underestimated the time needed to extend the parser in various ways that seemed important; but no doubt I am not the first linguist to make that mistake.

The aim also seemed worthwhile because it is so difficult to test a sophisticated generative grammar by hand. What I hoped to develop was a research tool - what I believe is called in Stanford a "grammarian's workbench" - which could be used in developing bigger and better WG grammars. Ideally, the same parser could be combined with a range of different WG grammars, including grammars for different languages as well as grammars containing different ranges of constructions and vocabulary in English. This need to keep the grammar and parser separate meant that the parser should not refer to any specific concepts contained in the grammar, though it necessarily referred to more general concepts. For example, it could refer to "dependent", but not to "subject"; and to "word", but not to "verb".

A consequence of the strictly academic aims of the project was that it was important that the parser should reflect the claims of the linguistic theory (WG) as closely as possible. In this respect it contrasts (in aims as well as achievements) with the larger-scale systems, which generally seem to be only loosely based on a linguistic theory. My belief was - and still is - that it should be possible to build a computer model of the processes by which the human brain analyses a sentence, and my aim was to get as close to such a model as was possible (given the limitations of my programming skills, of the available hardware and software, etc). The theory of WG includes some claims about processing (though it is primarily about the structures assigned), and since the aim of the project was to develop not only grammars but, through them, the theory itself, there would have been no point in building a computer system which disregarded these theoretical claims. A successful implementation would tell us nothing about the success of the theoretical claims about processing; nor would problems encountered in developing the implementation show anything about weaknesses in these claims.

The following are some examples of the effects of this principle of keeping the implementation as close as possible to WG theory:

a. Morphology had to be taken seriously, because WG claims that the general principle of default inheritance applies to morphology as well as to the rest of grammar. It would have been strange indeed to use inheritance in syntax but not to show how it applied to morphology, when many linguists seem to believe that it is more appropriate for morphology than for syntax. This meant that a system had to be developed for segmenting inflected words into stems and affixes, and for relating these patterns to morpho-syntactic features. Furthermore it had to be demonstrated that this system could also handle irregular inflections, since the regular rules only supply default values. This turned out to be an example of an area of WG theory that was less well developed than I had thought, so I had to work on the underlying theory as well as on the implementation of it.

b. Human processors clearly assign more than one interpretation to ambiguous words, even if they almost immediately eliminate most of them on the basis of context; and at least introspection supports the claim that most ambiguous sentences are handled by considering a range of alternatives in parallel, and closing most of them off quite quickly, rather than by considering one alternative at a time and back-tracking when the current one turns out to be a dead end (as in "garden-path" sentences). This claim means that the parser must be able to generate alternative analyses for the same word, and must carry these forward for consideration when later words are analysed. It also means, of course, that the analysis must proceed from left to right, one word at a time.

One practical consequence of these assumptions about the way in which the parser should handle ambiguities was that a satisfactory system had to be devised for naming the alternative readings of each word (and also of each string of words in a coordination). Thus instead of simply referring to flies in time flies like an arrow as "word 2", we must distinguish the noun reading from the verb reading, and we must also distinguish between two noun readings, according to whether flies is object of the verb time or head of the noun time.

Another practical consequence is that the system had to be designed to search for all possible analyses - a much slower process than searching for just one analysis. From an academic point of view, exhaustive analysis is essential in a grammar tester. If the system stops after it has discovered one analysis, then there is no way of knowing what other analyses might have been produced for the same string of words - an essential consideration when building a grammar. If the first, and only, analysis generated is the one the researcher hoped for, then it is very tempting to believe that all must be well.

Although the treatment of ambiguity loomed rather large, it was also made a lot easier by the academic goals of the research, because there was no obligation to resolve global ambiguities. The aim was to discover all the grammatical structures that were compatible with a given string, without trying to eliminate those that were silly (or incompatible with some hypothetical context).

c. The principle of using the implementation as a test for the theory meant that it was important to test as much of the theory as possible. Unfortunately there was no time even to start on the semantics, but I hoped to make progress not only in inflectional morphology but also in each of the two main areas of syntax: the grammar of **subordination**, and that of **coordination**. It seemed particularly important to cover both parts of syntax, however partially, because one of the most controversial claims of WG is that constituent-structure is the basis for coordination, but not for subordination. Outside coordination, relations among words are best described in terms of dependency, not in terms of constituents. The implementation in itself clearly could not show that this was the best way to treat grammar, but if successful it would at least show that this kind of treatment was feasible.

d. Implementing the theory by computer raised interesting possibilities for the relation between the human user and the theory. The question is what the structures of the data-base should be like, and more precisely, how complex they should be. So long as they have to be both produced and manipulated in the generation process by a human, they obviously need to be reasonably simple - not significantly more complex than a phrase-structure rule of the familiar kind, for instance. But why should we assume that the data-structures stored in our minds are actually as simple as that? It is at least conceivable that a parser would be more efficient, and nearer to psychological reality, if the data provided by the grammar was much more complex, and a computer implementation gives one the chance to explore this possibility.

Ultimately, of course, the linguistic facts have to be provided by a human, but it is possible to design the system in such a way that although the human input is manageable and easy, this input is converted into a more appropriate, and much more complex, form before being stored in the data-base to which the parser applies. The claim would then be that the more complex data-structures modelled psychological reality, although they were (psychologically) unmanageable for the human programmer/linguist, for whom the simpler structures were designed. It may of course turn out that the data-structures needed by the parser are in fact simple enough for a researcher to handle manually. If so, this will be a very interesting discovery. But we cannot know whether or not this is the case unless we can explore alternatives where it is not so.

To summarise, then, the aim of the project was to develop a parsing system as a tool which could then be used for developing WG grammars. It should be demonstrably capable of parsing constructions of widely different kinds, and of applying rules about inflectional morphology; ideally it would also have been able to add semantic structures for each word, but shortage of time prevented this. And the way in which it set about its task should bear as much similarity as possible to the way in which we might imagine a human doing it.

2. The implementation: technicalities

The implementation is written in Prolog2, and ran first on an IBM PC/XT, and more recently on an IBM PS2. The parser occupies about 130K of memory, and the working grammar about 80K. The latter is based on a "human" grammar which is 28K long. (The relation between these two grammars will be explained below.)

Prolog is built around propositions, which consist (as expected) of a predicate and a number of arguments. If a proposition is unconditionally true, it is called a "fact"; but if it is true only if one or more other propositions is also true,then it and its pre-conditions constitute a "rule", of which it is the "head". This distinction is important in the implementation, because the parser consists of nothing but rules, while the grammatical' data-base (like the grammatical structures generated) consists of nothing but unconditional facts. Furthermore, the predicates that occur in the grammatical data-base are drawn from a different vocabulary from those which occur in the heads of rules. This allows the grammatical database to be stored in a file which contains nothing else, while the parser is distributed (for convenience) among a number of other files. Thus the desired formal separation between grammar and parser is total.

To make the discussion somewhat more concrete, let us take a typical rule from the parser. This is the rule that is responsible for mapping the string of letters of which a word consists onto a "definition", a triple of (i) a set of morpho-syntactic features, (ii) a word-form, and (iii) a word-type; e.g. it will map the letters <fly> inter alia onto (i) [finite, imperative], (ii) base-form, (iii) flyV - i.e. the verb fly. Its head is a proposition whose predicate is "define" and which has two arguments: the name of a word (e.g. "word 1,3", meaning "the 3rd reading of the first word"), and a definition with the three parts listed above. When the rule is applied, the first of these arguments is already fixed, but the second argument is a variable. The effect of applying the rule is to instantiate this variable, by means of the various propositions whose truth must be verified before the head proposition is accepted as true - notably, another rule whose head-predicate is called "findModel", and whose role is to find some combination of features, word-forms and word-types which could be a model for the word being processed. This rule in turn defines other rules which must be satisfied, and which each instantiate one part of the final "definition" argument.

As may be imagined, the parser is a complex object which cannot be described here except in the most general terms (though copies are of course freely available on request). We shall comment below on some of its more general properties.

As explained earlier, the grammatical data consists entirely of Prolog facts. At one level of abstraction they are all similar, in that they consist of a predicate and just two arguments, but there are a variety of facts which have different statuses. These are quite easy to distinguish:

a. the grammarian's grammar,
b. the machine's grammar,
c. the sentence-word's observed properties,
d. the sentence-word's inferred properties.

These each allow a different range of predicates.

a. The grammarian's grammar:
X isa Y ("X is an instance of Y")
X his Y ("X [humanly] is Y")
X has Y ("X has Y")
X hisnot Y ("X, exceptionally, is not Y")
X hasnot Y ("X, exceptionally, has not Y"),

b. The machine's grammar:
nis (X,Y) ("X normally is Y")
but (X,andnot,Y,Z) ("X is Z but Y, exceptionally, is not"),

c. The sentence-word's observed properties:
ois (X,Y) ("X observedly is Y"),

d. The sentence-word's inferred properties:
iis (X,Y) ("X inferredly is Y").

From a theoretical point of view, the positive predicates of b-d are all the same. However it is essential to distinguish them in the implementation so that the computer's memory can be cleared of all "ois" and "iis" facts before each parse begins, in order to prevent the findings of different parses from getting mixed up together.

The propositions of b-d are exactly equivalent to a slot:filler notation such as is standard in the various PatrII implementations of GPSG, HPSG, LFG and so on in the unification tradition of parsers (Shieber 1985) or in McCord's Slot-grammars (e.g. McCord 1982). The notation thus allows a more direct comparison between WG analyses and those produced in better-known frameworks; and it captures the fact that every WG fact can be reduced to a combination of a variable and a value.

In contrast, the grammarian's grammar contains a wider range of predicates, and could contain a wider range still. These predicates are treated as operators, which means that they can be written, without commas, between their arguments, so the grammarian's input is in something approaching natural language (with added brackets and commas).

The following is a typical lexical entry (i.e. the set of propositions which refer to a lexical item), in "grammarian's format".

[1] [stem, of, wLikeV] his like.
[2] wLikeV isa lexicalVerb.
[3] wLikeV has [a, direct].

These three propositions define the properties of the lexical item whose name is "wLikeV" - i.e. the verb like. Proposition [3] says that wLikeV has a direct object; the other propositions are self-explanatory.

Some typical propositions about more general categories, again in grammarian's format, are given in [4-8].

[4] [features, of, determiner] his [features, of, [complement, of, itself]].
[5] lexicalVerb isa verb.
[6] word has [a, head].
[7] [finite, verb] has [no, head].
[8] [finite, verb] hasnot [a, head].

Proposition [4] deals with the agreement between a determiner and its complement (which, in the WG analysis, is the common-noun following it); it means "a determiner's features are the same as those of its complement". Propositions [6-8] illustrate the use of negative predicates to trigger overriding. The general rule [6] is that a word has a head, but finite verbs are an exception to this, because (according to the implemented grammar) they have no head. Proposition [7] is needed in order to guarantee that any finite verb inherits the property of having no head; but proposition [8] is needed to prevent such verbs from also inheriting the general property of having an obligatory head, as an alternative to the property in [7].

Propositions in grammarian's format are converted, by a special compiler which is distinct from the parser, into the machine's format. The grammarian's "isa" propositions are used to build complex names for the simple concepts referred to in the grammarian's other propositions. For example, the concept "lexicalVerb" is converted into the complex name "[speech,[word,[verb,[lexicalVerb,_]]]]"; and those grammarian's "his" propositions that define relations among features are similarly used for building complex names in terms of morpho-syntactic features. The remaining "his" propositions get translated directly into "nis" propositions (with suitable complication of the arguments by the above operations). And the grammarian's "X has [quantifier,Y]" propositions are converted into "nis" propositions whose first argument is "[quantity, of, [Y, of, X]]", and whose second argument is a pair of numbers which define the maximum and minimum number of examples of "Y of X" that are allowed. For example, if the quantifier is "a", the second argument is "[1,1]", meaning that both the maximum and the minimum are 1.

To recapitulate, then, the grammar is presented to the parser in "machine format", in which all propositions have one of the predicates "nis" and "but". This database is derived by the machine, before parsing starts, by a special program which applies to the grammar in "grammarian's format". This format is a compromise between the machine format and ordinary natural language, and in particular it allows a slightly wider range of propositions. These two formats are demonstrably intertranslatable, but it is possible that the machine's more complex format is closer to psychological reality than the other.

On the basis of this grammar, plus the parser, the implementation will accept a string of words, and produce, for each reading of each word in turn, a further set of propositions. One of these propositions for each word has the predicate "ois", and describes its observed property - namely, its spelling. The remaining propositions have the predicate "iis", and state facts about the word which can be inferred on the basis of the "ois" proposition. These two sets of propositions constitute the analysis of the word, but both their format and their number make them hard for the human grammarian to use; so they are converted into a more readable, summarised form on the screen. Some information is shown about the analysis of each word as it is being processed, and at the end of the string of words a simplified dependency structure is given for any coherent reading of the whole string.

3. Achievements: descriptive

This section describes the empirical coverage of the grammar. The number of constructions is rather small, but they include some of the constructions which are most difficult, and most interesting, from a theoretical point of view. The examples that illustrate each pattern below have all been tested on the implementation; however, I should admit to a few rather trivial idealisations, such as the fact that the implementation will not in fact accept capital letters or apostrophes.

3.1 Morphology

a. Spelling rules

Some affixes have alternative forms according to their orthographic context (e.g. the -s suffix may be spelt either <s> or <es>), and some suffixes require changes in the spelling of a stem - notably, the loss of "silent" <e>, the doubling of certain final consonants, and the replacement of <y> by <i>. All these changes are allowed by general rules.

(1) cats, potatoes
(2) like, liking
(3) big, biggest
(4) fly, flies

b. Word-forms

The relations between stems and affixes on the one hand, and morpho-syntactic features on the other, are mediated by the notion "word-form" - e.g. the "en-form" of a verb is mapped onto the features for either the perfect or the passive participle. Some word-forms, for some words, are irregular, and are indicated as lexical exceptions to the general rules.

(1) cats, sheep, *sheeps
(2) liked, flew, *flied

A weakness of the implementation is that all irregular stem-forms are treated as though they were suppletive; a better treatment would show, for example, that <fly> and <flew> share the same consonants and differ only in their vowels.

Some distinctions are shown only in terms of word-forms, without reference to features. This is the case for any distinction that is made only for a lexically restricted set of words. Particularly clear examples of this in English are (i) the distinction between subject and non-subject forms found in some personal pronouns, and (ii) the distinctions which are made only by the verb be.

(3) she, her
(4) am, are
(5) was, were

The word-forms covered include those formed by the addition of n't to a tensed polarity ("auxiliary") verb.

(6) wasn't, can't

c. Morpho-syntactic features

Feature-analyses are given to all words, on the basis of their word-form plus their word-type. Thus the s-form of a noun is given a different set of features from that of a verb. Appropriate feature analyses are given to all the word-forms listed below. Where a word-form is given more than one such analysis, the number of alternatives is indicated in brackets after the relevant example.

(1) cat, cats
(2) soon, sooner, soonest
(3) like [3], liking, likes, liked [3]

However, the word-type of a word includes not only its classification in terms of general word categories (noun, verb, etc), but also its identity as a lexical item, so lexical irregularities of feature assignment can be accommodated. These irregularities include unexpected features (e.g. linguistics, although an s-form noun, is singular) and feature-gaps (e.g. oats has no singular, must has no past, and the irregular form aren't cannot have features which allow it to follow its subject, as in *<u>I aren't good enough</u>). Other feature-gaps have more general explanations - e.g. modal verbs must be tensed (e.g. *<u>canning</u>), and a noun cannot be both mass and plural (hence *<u>sunshines</u>).

A serious gap in the empirical coverage is the absence of any analysis for gerunds. These are problematic in WG for the same reasons as in other theories, and require further research.

d. Class-changing derivational morphology

One class-changing affix is included: the suffix -<u>ly</u>, which changes an adjective into an adverb (e.g. <u>quietly</u>). Although the lexical information comes from an adjective, the result is classified as an adverb, indistinguishable from any non-derived adverb.

3.2 Syntax

e. Adjuncts

Common nouns are allowed to have any number of adjuncts either before or after them, but preceding and following adjuncts are distinguished: the former may be either an adjective or a singular common noun, the latter may be a preposition.

(1) small cat, small grey cat
(2) box lid, tray box lid
(3) cat in a tray in the sunshine

If both a noun and an adjective are adjuncts of the same head, then the adjective must precede the noun:

(4) small box lid, *box small lid

Verbs may have any number of following adjuncts, which may be prepositions or adverbs.

(5) She purrs quietly in a tray in the sunshine.

If the same word has both a complement and a following adjunct, then the complement must come first.

(6) She washes the dish quietly. *She washes quietly the dish.

f. Complements

Only a small range of complement-patterns is allowed for verbs:

- a direct object,
(1) He likes her. *He likes. *He likes greatly.

- indirect + direct object,
(2) He gave the cat a tray.

- a predicative (which may be a present participle),

(3) She is small. She is purring.

- a non-finite complement,
(4) She has been purring.

- an otherwise unspecified complement.
(5) She is Smudge. She is in it.

Prepositions are also allowed to have an otherwise unspecified complement, as illustrated by (5).

According to the WG analysis, a determiner (a subtype of pronoun, and therefore of noun) also has a complement, the following common noun. These two words agree with one another in terms of their morpho-syntactic features.

(6) this chocolate, *this chocolates, these chocolates, *these chocolate

The agreement concerned works strictly in terms of dependency relations, not linear order; so it "skips" any noun between the determiner and its complement.

(7) these chocolate boxes, *this chocolate boxes.

One further restriction on common nouns is that a singular countable common noun cannot occur without a determiner as its head.

(8) He likes sunshine. *He likes lid.

g. Subjects

A subject is obligatory with a tensed verb:

(1) She purrs. *Purrs.

What is traditionally called "agreement" between the subject and verb is not treated in the same way as the (genuine) agreement between determiners and common-nouns discussed above. Instead, it is taken as a restriction imposed by the subject on the verb, provided the latter has the feature "subject-agreement" - i.e., if the latter is in the present tense.

(2) She purrs. *They purrs. *She purr. They purr.

The effect on the verb varies according to whether or not the subject is coordinate.

(3) Tom and Dick are. *Tom and Dick is.

And it also varies lexically from subject to subject, I and you being exceptions to the general rule that a singular subject takes a verb with -s.

(4) I purr. *I purrs.

Extra lexical restrictions are also imposed by the verb be:

(5) I am. *I are. *They am.
(6) I was. He was. *They was. *You were.
(7) They were. You were. *I were. *He were.

A further restriction on subjects applies to the handful of personal pronouns whose form varies between subject and non-subject position.

(8) She purrs. *Her purrs.
(9) I like her. *I like she.

Although subjects normally precede their predicates, in the case of tensed polarity ("auxiliary") verbs the subject may either precede or follow.

(10) She is. Is she? Has the cat been purring?

In one case it is not optional, but obligatory, for the subject to follow its verb, namely if the verb is aren't; as explained earlier this is treated in terms of a constraint on the combinability of features, the verb's features being different according to whether its subject precedes or follows it.

One particularly tricky pattern of restrictions arises if an inverted subject is coordinate, as in (11).

(11) Are Tom and Dick?

This is difficult because the natural point at which to check the form of the verb is during the processing of Tom, but at that point there is no evidence that the subject is coordinate. It is easy to see how to solve the problem, but the solution looks rather expensive and has not been included in this implementation (which therefore rejects (11)).

h. Negatives

As already noted, verbs that contain the suffix -n't are handled by the morphology, but those with not as a dependent must be dealt with by syntactic rules. According to our grammar not is not possible for lexical verbs:

(1) She is not. *She purrs not.

The not always follows the verb, and cannot be separated from it by a complement verb.

(2) She is not. *She not is.
(3) She is not purring. *She is purring not.

However, if the verb's subject is postposed, it must precede not.

(4) Is she not purring? *Is not she purring?

i. Shared dependents

One of the characteristics of WG analyses, in contrast with other dependency analyses, is that under certain conditions a word may have more than one head. The least controversial case is that of shared subjects, where the treatment is the same as in some constituency-based theories, such as Lexical-Functional Grammar and Head-driven Phrase Structure Grammar. Thus if a verb is either a subject-raising verb, or a verb of control, then its subject is also the subject of its complement. In our grammar, the only examples of such verbs are the polarity verbs (be, have, can), so it can best be illustrated with examples like (1).

(1) She has been purring.

Here she is analysed as the subject not only of has, but also of been and of purring.

Another construction in which, according to the WG analysis, a word can have more than one head is the one responsible for extraction, as in topicalisation and wh-movement. Topicalised objects are allowed, as in (2).

(2) Smudge Dick is washing.

Here the topicalised object of washing is Smudge, which is also a dependent of Dick. However, the treatment of extraction in the implementation has only just started, and several refinements are needed.

j. Coordination

All the basic patterns of coordinate structures, with and without layering, are allowed; so the following examples all receive the correct number of structures (shown in brackets).

(1) Tom, Dick and Harry [1]
(2) Tom and Dick and Harry [3]
(3) Tom and Dick or Harry [2]

The conjuncts need not be single words.
(4) a box and a lid
(5) small boxes and lids
(6) small boxes and big lids
(7) very big and small cats

The roots of the conjuncts share their external relations.
(8) He likes Tom and Dick.
(9) Tom and Dick flew.

But there is no requirement that each conjunct should have just one root - in constituency terms, that each conjunct should be a complete constituent.
(10) He likes Tom greatly and Dick slightly.

The analysis even extends to rather simple gapped constructions.
(11) Tom likes Dick and Dick, Tom.

4. Achievements: theoretical

Various changes to the theory suggested themselves during the time when the implementation was growing, but it is hard to be sure which of these arose from the computational work and which had other origins. The following are probably the main theoretical developments which are most closely connected to the work on the implementation.

a. Morphology

The notions "word-form" and "type" are the main innovations here. As explained earlier, a word's word-form is a property which mediates between its morphological structure (in terms of stems and affixes) and its morpho-syntactic features, and which is of particular value in dealing with syncretism. A word's type is the general class of which it is an instance - as _cat_ is an instance of "common noun", for example. This is relevant to morphology because of class-changing derivational morphology - the relation between _quiet_ and _quietly_, for example. The relevant generalisation here is that the type of the ly-form of an adjective is "adverb". Such examples raise challenging problems for any theory like WG which uses isa hierarchies as the basis for generalisations, because _quietly_ needs to be able to inherit some of the properties of _quiet_, whilst having a different type.

b. Formal properties of propositions

The possibility of converting the grammarian's format automatically into the machine's format, in which all propositions have a single predicate, has shown that we cannot claim that knowledge must be represented in terms of a variety of predicates. The formal properties of the notation may or may not be an empirical matter, but this work has suggested that it is probably not a crucial difference between WG and other theories.

c. Inheritance and overriding

A good deal of effort had to be spent in working on the system for inheriting information down isa hierarchies, and for blocking this in exceptional cases. The most important conclusion of this work is probably that overriding should be stipulated, rather than automatic, because otherwise it is not possible to allow alternative values for the same variable. This is the function of the grammarian's negative predicates, "hisnot" and "hasnot", which convert into the machine's "but" predicate.

However, another conclusion also arose from experimenting with the difference between grammarian's and machine's notation: that concepts could, and perhaps should, be given complex names which simultaneously distinguish them from other concepts, but also locate them in an isa hierarchy (Fraser and Hudson, in preparation). This conclusion is not quite as certain as the previous one. For example, the possibility of class-changing word-formation rules mentioned earlier suggests that the type should be kept distinct from its name. Moreover, there are words which, in the grammar-lexicon, should be shown as having ambiguous status in the hierarchy of word-types - e.g. dare is either a lexical verb or a polarity verb - but which have a lot of properties in common in both cases. This is hard to express if the name of the concept automatically locates it uniquely in an isa hierarchy. A great deal more thought is needed in this area, but the question would almost certainly not have arisen outside a computational context, because of the practical difficulties in writing long and sometimes complex structured names by hand.

d. Adjacency

One of the foundations of WG is the Adjacency Principle, which states that the phrase consisting of all the subordinates of any given word must be continuous (with the important exception of those cases where a dependent also has another head). This principle has a rather ambiguous status, because it is not clear whether it is part of the knowledge-base or of the processing system. This ambiguity persists, but it is clarified somewhat in the implementation because it turns out that the latter does not need to contain the Adjacency Principle as such. Rather, the processing system - i.e. the parser - is organised in such a way that the Adjacency Principle is automatically respected.

To be specific, the parser makes use of the following algorithm for finding dependents and heads of the current word W:

1. Try to take the nearest preceding word X that has no head as a dependent of W.
    a. If successful, repeat 1, with reference to the last word before X that has no head;
    b. Otherwise, go to 2.
2. Try to take a root of the nearest preceding word Y as head of W.
    a. If successful, stop.
    b. Otherwise, go to 3.
3. Try to take W as a word which need not have a preceding head, either because it needs no head at all, or because it may have a following head.
    a. If successful, stop.
    b. Otherwise, go to 4.
4. Try to take W as the root of a conjunct which shares its external relations with earlier conjunct-roots of a coordination.
    a. If successful, stop.
    b. Otherwise, fail.

This algorithm in fact guarantees that the dependency structures which are identified will respect the Adjacency Principle (with appropriate provision made for dependencies shared in coordinate structures). But the output of the implementation would not be changed if we added a rule which tested every dependency structure for "adjacency", because every structure is bound to respect it.

The implementation has thus clarified the situation by showing that the Adjacency Principle may not in fact be needed in the grammar, because it is a consequence of the way in which the parser is structured. If it can be shown that the system for producing sentences is also structured in such a way as to respect the Adjacency Principle then we may conclude that it is certainly not part of the grammar. However, it is noticeable that it is functionally just like any fact in the grammar: a generalisation about permitted grammatical structures. This confirms that although the parser and grammar can be distinguished formally, they cannot be distinguished so clearly in terms of their respective functions.

The general conclusion, then, about the relations between the computer implementation and the underlying theory, is that the implementation has pointed to some ways in which the theory needed to be changed, but has also offered new possibilities for theorising which might not otherwise be considered (such as the possibility of internally structured names). In either case, then, the implementation has had an important heuristic function.

5. Plans for the future

The implementation described above is a first step, but no more than that. Its main contribution to linguistics probably lies in the foundations that it provides for further work rather than in any of the achievements listed here. This future work could follow a number of different paths.

Most obviously, we could expand the grammar for English. We could provide a much bigger and richer vocabulary, and/or we could fill the many serious gaps in the general grammar. Some of these gaps could be filled quite straightforwardly, on the basis of the WG analyses that are already available - for relative clauses, reported clauses, island constraints on extraction, passives, extraposition and possessive constructions, for example. Other gaps represent gaps in WG analyses which remain to be filled, but which promise to be fairly straightforward - e.g. comparative constructions, tag questions, reported speech, delayed subjects. It is at least possible that all these constructions, and extra vocabulary, could be added without any changes at all to the parser. Other gaps again represent serious challenges to the theory, notably the analysis of gerunds mentioned earlier. It seems likely that these will eventually lead to some changes in the theory.

Another obvious direction for future work is in expanding the coverage to languages other than English - the more different, the better. The parser is designed in principle to apply equally to all languages, but of course there are some patterns that are more prominent in other languages than in English (e.g. cliticization, complex inflections, incorporation) and which would need a somewhat different parser.

One kind of development which would certainly need additions to the parser would be any attempt to add semantic analyses. A good deal of this work could already be done quite easily, on the basis of existing WG theory, though a major descriptive and theoretical problem will arise early on in this work, namely the need for a vocabulary of semantic relations. No doubt other important problems would also present themselves, and it is hard to predict how far the existing grammar and parser would need to be changed, apart from simple additions, in order to accommodate semantic structures. One thing that is fairly certain is that such work would quickly turn into research on general knowledge structures rather than on semantic structures proper, but serious work on general knowledge is especially necessary in a theory which lays as much emphasis on the similarities between linguistic and non-linguistic knowledge as WG does.

Another avenue for further research is at a more theoretical level. While the parser was being developed, my ideas on WG theory continued to change - often as a direct result of the computational work, as I suggested above. I was able to embody some of these changes in the developing parser, but others were too radical to incorporate without undoing large parts of the parser that had already been built.

Finally, the parser itself needs a thorough reworking. As so often happens, the system grew piece-meal, which meant that many generalisations had to be missed - though equally, some generalisations are captured fairly satisfactorily; for example, a single clause, "findDependent", is used for working out the dependency relation between any two words, whether the candidate for dependent comes first or second. The same cannot be said of other parts of the system, where different operations involve essentially the same sub-goals. It would be good to redesign the system in the light of the work done so far, in the hope of moving it towards the ideal goal of a very short parser consisting just of a few extremely general operations, equally applicable to linguistic and to non-linguistic knowledge. If one ignores the details, it is already clear what the outlines of such a parser should be; but of course one cannot ignore the details in a computer implementation.

Acknowledgements

The development of the computer system described here, between the autumn of 1987 and late 1988, was made possible not only by a personal research grant (C 00 24 2022) from the ESRC but also by the very generous support of the Head of Department, Professor Neil Smith. The hardware and a great deal of technical help were provided by Dr Richard Sharman, of the IBM(UK) Scientific Centre at Winchester, and Norman Fraser also gave me a lot of help, in the form of expert advice, help in thinking issues through, and encouragement. I am deeply grateful to all three of them.

References

Fraser, N. M. (1985) A Word Grammar Parser. University College London: MSc dissertation.

Fraser, N. M. (1987) A Word Grammar Parser. Progress report 1. University College London. mimeo.

Fraser, N. M. and Hudson, R. A. (in preparation) Inheritance in Prolog. mimeo.

Grantham, P. R. (1987) Natural Language Understanding. A Word Grammar approach to the problem. Sheffield City Polytechnic: MSc dissertation.

Hudson, R. A.(1984) Word Grammar. Oxford: Blackwell.

Hudson, R. A.(1985) The limits of subcategorisation. Linguistic Analysis 15. 233-255.

Hudson, R. A.(1986a) A Prolog implementation of Word Grammar. Speech, Hearing and Language: Work in progress, 2. University College London: Phonetics and Linguistics Dept. 133-150.

Hudson, R. A.(1986b) Sociolinguistics and the theory of grammar. Linguistics 24. 1053-1078.

Hudson, R. A.(1987a) Zwicky on heads. Journal of Linguistics, 23, 109-132.

Hudson, R. A.(1987c) Grammar, society and the pronoun. In Steele, R. and Treadgold, T. (eds) Language Topics. Essays in honour of Michael Halliday. Benjamins. 493-505.

Hudson, R. A.(1988a) Coordination and grammatical relations. Journal of Linguistics 24. 303-342.

Hudson, R. A.(1988b) The English passive. distributed by LAUD.

Hudson, R. A.(1988c) Extraction and grammatical relations. Lingua 76. 177-208.

Hudson, R. A.(1989a) Identifying the linguistic foundations for lexical research and dictionary design. International Journal of Lexicography 1.3. 287-312.

Hudson, R. A.(1989b) Gapping and grammatical relations. Journal of Linguistics 25.

Hudson, R. A. (forthcoming a) Recent developments in dependency theory. In J. Jacobs, W. Sternefeld, T. Vennemann and A. von Stechow (eds.) Syntax. An International Handbook of Contemporary Research. Berlin: de Gruyter.

Hudson, R. A. (forthcoming b) Language, metalanguage and conceptual structure. In F. Heyvaert and F. Steurs (eds.) Words Behind Worlds. Leuven: University of Leuven Press.

McCord, M. C. (1982) Using slots and modifiers in logic grammars for natural languages. Artificial Intelligence 18. 327-367.

Shieber, S. (1985) An Introduction to Unification-based Approaches to Grammar. Stanford: CSLI.