Inheritance in Word Grammar

Norman M. Fraser[*]
University of Surrey

Richard A. Hudson[†]
University College London

This paper describes the central role played by default inheritance in Word Grammar, a theory of language knowledge and processing. A single formalism is used to represent knowledge at the levels of morphology, syntax, and semantics. A single rule of inference is used to inherit knowledge at all of these levels. This rule is distinctive in that it requires defaults to be explicitly overridden in the case of exceptions. The explicit overriding rule is used in syntax to achieve what other theories achieve by means of transformations, metarules, or lexical rules.

1. Introduction

Since the scientific study of language first began, a central concern of linguists has been the identification of linguistic generalizations and, where necessary, the stating of exceptions to these generalizations.[1] However, it is only within the last few years that linguists have begun to think of this process in terms of the construction of default inheritance hierarchies. This new way of envisaging old problems is attractive for at least three reasons. Firstly, it encourages linguists to be explicit not just about the relations that hold between individuals and classes, but also about the relations that hold between different classes. For example, where the nouns of a language have traditionally been assigned to some number of distinct morphological paradigms, the default inheritance approach encourages the morphologist to pay attention to generalizations that cut across paradigms. If these generalizations are inherited, then there must be some shared super class to inherit from, and the system of word classes and paradigms must be designed accordingly.

Secondly, whereas generalizations have traditionally been class-based, in the inheritance approach they are based on typical cases and their features, any of which may be overridden. Thus the shading from core members of a class to peripheral members can be accommodated—indeed, the existence of peripheral members is predicted by the mechanism for overriding defaults. The third and more pragmatic reason why it is useful to recast well-known linguistic problems in terms of default inheritance is that there is a fairly well-developed—though by no means conclusive—body of knowledge on the subject in the artificial intelligence field of knowledge representation (e.g. Etherington and Reiter 1983; Brachman 1985; Touretzky 1986; Etherington 1988). Nearer the computer science mainstream, work in object-oriented programming languages (Cook 1989) offers an interesting range of relevant insights and inheritance-based tools.

[*] Social and Computer Sciences Research Group, University of Surrey, Guildford, Surrey, GU2 5XH, United Kingdom. E-mail: norman@soc.surrey.ac.uk.
[†] Department of Phonetics and Linguistics, University College London, Gower Street, London, WC1E 6BT, United Kingdom. E-mail: r.hudson@ucl.ac.uk.
[1] We received very helpful comments on an earlier draft of this paper from three anonymous *Computational Linguistics* readers, to whom we are most grateful. We also benefitted from discussions with participants at the Workshop on Inheritance in Natural Language Processing, Tilburg, August 1990.

In recent years, linguists and computational linguists in particular have begun to explore problems at most linguistic levels within inheritance frameworks. For example, Gibbon and Reinhard have proposed inheritance-based solutions to problems of phonology and prosody (Gibbon 1990; Reinhard and Gibbon 1991). Most work to date has centered on morphology (e.g. De Smedt 1984; Flickinger, Pollard, and Wasow 1985; Daelemans 1987; Calder 1989). A certain amount has also been achieved in syntax (e.g. De Smedt 1984; Flickinger 1987), where inheritance is used to construct subcategorization frames for words. As for semantics, there has been a great deal of work on inheritance in so-called 'semantic networks,' but much of this work relates only loosely to the semantics of natural language. The work we present in this paper differs from all previous work in natural language processing (NLP) in at least two respects. Firstly, it is distinctive in the *extent* to which inheritance is used. Within our framework knowledge at all levels (morphology, syntax, semantics, world knowledge) is integrated in a single inheritance hierarchy. Indeed, given the extent of integration, some of these level distinctions must be regarded as arbitrary. Secondly, it is distinctive in the *purposes* for which inheritance is used. The canonical application of inheritance in NLP is lexicon construction. Our system uses inheritance for this purpose but it also makes inheritance play a vital role in the building of structure during parsing.

What we describe is part of a theory of language (knowledge and processing) called Word Grammar (WG) (Hudson 1984; 1990). Section 2 introduces the knowledge representation language used in WG. Section 3 outlines the use of inheritance in WG to describe the facts of syntax and semantics. Concluding observations are drawn in Section 4. An Appendix sets out a fragment of English grammar and a simple sentence analysis.

2. Word Grammar

In this section we define the syntax of WG propositions and explain how they can be interpreted. The Appendix contains a fragment of English grammar from which all examples are drawn.

2.1 Propositions

One of the central claims of WG is that knowledge of language is a sub-component of knowledge in general, and in particular that it is a kind of propositional knowledge (which we assume must be distinguished from other kinds of knowledge, notably perceptual knowledge). This amounts to the rather uncontroversial claim that all linguistic knowledge may be expressed in terms of propositions (just as it can be expressed in terms of attribute-value structures). This is uncontroversial because it is obviously possible to represent any standard linguistic structure or rule as a collection of propositions, though the same is probably not true for representations of faces, sounds and so on, which are based more directly on perception. The use of propositions to represent linguistic knowledge allows us to use standard logical operations as the basis for parsing. One set of propositions defines the observable properties of the input, and parsing consists of drawing inferences. These inferences constitute the analysis of the input, so, as in unification-based systems, the formal properties of sentence structure are the same as those of a grammar.

Propositions are of two types, positive and negative. A positive proposition consists of a predicate and two arguments. By convention, an infix notation is used:

(1a) noun isa word.

(1b) verb has (1 finiteness).

(1c) (stem of JUMP) = <jump>.

The parentheses are generally redundant, so in later examples we shall omit them. As we shall see, the arguments are usually somewhat more complex than in these examples.

A negative proposition consists of 'NOT:' followed by another proposition:

(2a) NOT: tensed verb has 0 subject.

(2b) NOT: position of predependent of word = after it.

A negated proposition may itself be negated:

(3) NOT: NOT: position of subject of v+s verb = after it.

Negated propositions play a crucial role in the WG system for default inheritance, as we shall explain below.

2.2 Predicates

Three different predicates are recognized:[2] 'isa,' '=,' and 'has.'

The 'isa' predicate is used to encode the relationship between a subtype and a supertype in a type hierarchy. This predicate is used to express both the relationship between instances and types (such as the well-known relationship of 'Clyde' to 'elephant') and the relationship between types and supertypes (such as the relationship of 'elephant' to 'mammal'). Instances and types are collectively known as **concepts.**

(4a) Clyde isa elephant.

(4b) elephant isa mammal.

The '=' predicate indicates identity of arguments. The reason it is necessary to include this predicate is that it is possible to identify the same concept by means of different kinds of names. For example, assuming that Clyde is grey, we can identify the concept 'grey' either by its atomic name or as a function of Clyde. The '=' predicate shows this identity. More complex identifications are of course also possible.

(5a) color of Clyde = grey.

(5b) mother of Clyde = sister of father of Babar.

(5c) mother of mother of Clyde = mother of father of Babar.

2 During the evolution of WG this figure has varied between five (Hudson 1984) and one (Hudson 1989), although the expressiveness of the formalism has not changed significantly. There is a balance to be struck between having a minimal (i.e. one-member) predicate set with necessary distinctions marked in the arguments, and having a more readable notation that includes extra predicates. The three-member set has been used in recent WG publications. In at least one computer implementation (described in Hudson 1989) the 'grammarian's grammar' is written with three predicates and compiled into a 'machine grammar' that uses only one predicate.

Propositions including the '=' predicate can be thought of as path equations of the sort used to indicate coreference in directed acyclic graphs.

The 'has' predicate is used to associate attributes with concepts. A proposition of the form shown in (6) associates concept Y with concept X in the quantity specified in Q.

(6) X has Q Y.

Q is called a **quantitator.** It signifies how many instances of the specified type should be associated with X.

(7a) word has 1 head.

(7b) finite verb has 0 head.

In the simplest cases, as in most of the examples in this paper, the quantitator can be just a single integer, but it is also possible to specify a range of numbers by giving the minimum and maximum, e.g. [0–1].

2.3 Arguments

Arguments fall into nine basic types. It is helpful to be able to describe these types in respect of their structure without reference to their function as arguments. We shall therefore say that a well-formed argument must be a **name** where a name conforms to one of the following definitions.

Atoms. The atoms of the WG knowledge representation are single words such as 'verb' or hyphenated words such as 'proper-noun' that identify single nodes in the knowledge structure.

Sets. A set of concepts is enclosed in set brackets. The first element inside the brackets is a sign that identifies whether conjunction or disjunction is intended. '{&: A, B}' means 'A and B.' '{/: A, B}' means 'A or B.' Special notations are used for two particular kinds of ordered 'and' set. Strings of orthographic symbols are enclosed in angle brackets (e.g. '<did>'). Linear constituent structures are formed by linking constituents by '+' (e.g. 'stem of it + mEd').

Relational names. These consist of an atom that is the name of a relation, followed by 'of', followed by a name that may be another relational name. Thus, 'A of B' and 'A of B of C' are both well formed (e.g. 'subject of verb,' 'position of subject of verb'). Relational names are right-embedding.

Positional names. These consist of positional atoms ('before,' 'after,' 'adjacent-to,' 'next-to') followed by a name, e.g. 'before X.' Positional names identify positions in a linear sequence in which the named concept is located.

It. Where a proposition refers to the same concept on either side of a path equation, the second instance of the concept is identified by the name 'it.' Example (8) uses 'it' to refer to 'word.'

(8) position of dependent of word = after it.

The concept identified after the '=' by 'it' is always identified before the '=' by the most deeply embedded (i.e. rightmost) name, which in this example is 'word.'

Compound names. These consist of two parts, the first of which is the value of a feature or a set of feature values, and the second of which is an atom. Thus 'past verb' and '{&: past, positive, s+v} polarity-verb' are well-formed compound names.

Temporary names. Stored concepts in the WG knowledge structure are **types**; e.g. 'noun' means 'the typical noun,' and 'subject of verb' means 'the typical subject of the typical verb.' To distinguish particular **tokens** from these types, tokens are assigned temporary names as they are encountered during processing. These names are temporary in the sense that they do not belong to the permanent knowledge base, instead being introduced during processing. Temporary names consist of integers prefixed by a character or characters. By convention, morpheme instances are prefixed by 'm,' word instances by 'w,' and objects in the semantics by 'c' (for 'concept'). Thus 'm1,' 'w12,' and 'c6' are all well-formed temporary names. For example, the propositions in (9) (which refer to the sentence analyzed in the Appendix, *Mary jumped*) illustrate the use of temporary names.

(9a) whole of w1 = <Mary>.

(9b) whole of w2 = <jumped>.

(9c) position of w1 = before w2.

The analysis of a sentence involves taking the observable facts, such as the above, and inferring unobservable ones which are logically consistent with them, with each other and with the knowledge base. The inferred facts for *Mary jumped* include the following:

(10a) w1 isa MARY.

(10b) w2 isa past JUMP.

(10c) subject of w2 = w1.

(10d) w2 has 1 subject.

(10e) position of subject of w2 = before it.

(10f) sense of w2 = c1.

Instance names. These consist of a name preceded by 'a' (or 'an'). Whereas temporary names provide a means of identifying specific instances, instance names provide a means of identifying any single instance of a specified type. If a type 'X' exists in the inheritance hierarchy, then the instance name 'a X' refers to any instance that isa X. For example,

(11) subject of passive verb = a complement of it.

Notice that without this 'a,' a name refers to *every* example of the type concerned. (This follows from the interpretation of concepts as types; if something is true of some concept, then it must also be true of every instance of that concept, barring specified exceptions). Proposition (11) identifies the subject of a passive verb with just one of its complements; without 'a,' it would identify the subject with every one of the complements, and lead to chaos.

Quantified names. As already explained, these consist of a quantitator followed by an atom: e.g. '1 head,' '0 complement.'

2.4 Interpretation

The WG knowledge representation language keeps close to ordinary English, as can be seen from the examples given above and from the Appendix. It avoids the ambiguity of ordinary English, and is much less rich, but the propositions are easy to understand. A full formal account of the semantics of the language would require a separate paper; in this section we limit ourselves to a brief discussion of one of the simpler areas, viz the use of positional names.

Positional names, which it will be recalled consist of a word such as 'before' or 'after' followed by another name, are primarily to do with relations in time—the relations between co-occurring spoken words, between co-occurring phonemes, or between events or times that are referred to in the semantic structure of a sentence. If the data to be analyzed are written, then some of these relations are mapped onto the spatial patterns of writing. With this reservation, then, 'before X' is the name of some time earlier than X, where X is itself either a time (say, last Friday) or an event that can be located in time (e.g. Mary's birthday party). The examples in (12) illustrate two uses of this single general pattern.

(12a) position of dependent of word = after it.

(12b) time of referent of past verb = before it.

The first example refers to the order of words in a sentence, while the second refers to the deictic relation between the event referred to by a past-tense verb and the time when that verb itself is uttered. (In (12b), 'it' is coreferential with the verb, so when this proposition is inherited by a token of a past verb, 'it' refers to this token, or more precisely to the time when it is uttered).

The rule for interpreting 'after' or 'before' must therefore be capable of determining which of two events occurs before the other. This is straightforward if these events are themselves given temporary names whose integer rises with time; w2 occurs, by definition, before w3, and c2 before c3. In this way, all temporary concepts are effectively time-stamped. All the rule needs to do is compare the integers.

Positional names are noteworthy because they illustrate particularly clearly the extent to which different kinds of knowledge can be integrated into a single system. The same formal apparatus, interpreted by the same inference rule, is used in syntax (regarding word order) and also in semantics (regarding temporal ordering of events). Moreover, the latter events themselves include not only the events referred to (e.g. the event of Mary jumping), but also the event of uttering the words concerned (i.e. in *Mary jumped*, the utterance of the word *jumped*).

2.5 Inheritance

Inheritance is the rule of inference that derives new propositions from existing ones. It is sanctioned primarily by any occurrence of the 'isa' predicate, but also by various other formal patterns mentioned below. If A isa B, then A inherits all the properties of B (except those that are blocked as we explain in the next section). In other words:

Inheritance. If A isa B, then for any true proposition P that refers to B, it is possible to infer another true proposition Q that is the same as P except that A is substituted in Q for B in P.

For example:

(13a) Clyde isa elephant.

(13b) color of elephant = gray.

(13c) color of Clyde = gray. [from (13a,b)]

A similar interpretation applies to the '=' predicate, which is in effect a reciprocal 'isa'. If A = B, then any true proposition that contains A can be matched by another in which B replaces A, and vice versa. Inferentially, then, both 'isa' and '=' are extremely simple, and extremely powerful, allowing the creation of new propositions by simple substitution operations.

The most noteworthy feature of the WG inheritance system is, once again, that it applies to all types of knowledge, allowing a single integrated knowledge base and a single set of inference rules for both linguistic and nonlinguistic knowledge (cf. the 'preference rules' of Jackendoff 1983). The same rule that allows us to inherit information about Clyde also allows us to inherit information about words within the grammar and about words in sentences. These similarities can be seen from the following examples.

(14a) noun isa word.

(14b) word has 1 head.

(14c) so: noun has 1 head.

(15a) MARY isa noun.

(15b) noun has 1 head. [= 14c]

(15c) so: MARY has 1 head.

(16a) w1 isa MARY.

(16b) MARY has 1 head. [= 15c]

(16c) so: w1 has 1 head.

These examples show how inheritance allows information to be inherited both within the grammar (14, 15) and from the grammar to a particular sentence word, in the process of parsing (16). As already explained, the aim in parsing is to link each sentence word to a word in the grammar from which it can inherit a set of propositions compatible with the propositions inherited for all the other words in the same sentence.

Another set of examples applies the same inheritance rule to meanings and concepts:

(17a) jumping isa action.

(17b) action has 1 actor.

(17c) so: jumping has 1 actor.

(18a) sense of JUMP = jumping.

(18b) jumping has 1 actor. [= 17c]

(18c) so: sense of JUMP has 1 actor.

(19a) w2 isa JUMP.

(19b) sense of JUMP has 1 actor. [= 18c]

(19c) so: sense of w2 has 1 actor.

(20a) w2 isa word.

(20b) referent of word isa sense of it.

(20c) referent of w2 isa sense of it.

(21a) referent of w2 isa sense of it. [= 20c]

(21b) sense of w2 has 1 actor. [= 19c]

(21c) so: referent of w2 has 1 actor.

If we continue this chain of deductions we eventually find that Mary is the actor of the event of jumping referred to by w2; in other words, Mary jumped. If the analysis were embedded in a body of knowledge about the world in which Clyde trod on Mary's toes, then we could infer that the person on whose toes Clyde trod jumped; and so on.

The unified nature of inheritance in WG allows us to recognize, or at least imagine, a single inheritance hierarchy for the whole of knowledge, within which linguistic concepts can be located as special cases of more general ones. In particular, words are a special kind of action, and inherit from 'action' properties such as having a time and an actor:

(22a) word isa action.

(22b) action has 1 actor.

(22c) so: word has 1 actor.

(22d) action has 1 time.

(22e) so: word has 1 time.

It was this inheritance that allowed us to assume that a word has a time, which can be referred to not only in the rules for word order but also in those for the semantics of tense (cf. (12) above).

Another direction in which WG extends the normal scope of inheritance is by allowing it to apply to relations as well as to the more familiar kind of nonrelational category, such as elephant, word, etc. (For a similar approach see Thomason and Touretzky 1991). This allows us to recognize a hierarchy of grammatical relations, with, for example, 'object' as a particular kind of 'dependent'; which in turn allows us to formulate word-order rules that refer to the appropriate point in the hierarchy, and then automatically generalize to all relations below this. Here is a simple example of the inferences that can be drawn.

(23a) position of dependent of word = after it.

(23b) object isa dependent.

(23c) LIKE isa word.

(23d) so: position of object of LIKE = after it.

To summarize, then, inheritance plays a much larger part in WG than in other theories. It allows us to locate atomic concepts in inheritance hierarchies, and encourages us to try to unify them all into a single grand hierarchy that reveals the continuities between linguistic and other concepts. A fortiori, it integrates linguistic categories of different levels into a single system, in which the same inheritance rule applies to morphology, syntax, and semantics. Moreover, since WG uses dependency instead of constituent structure, all the units of syntax (outside coordination) are single words, so the only difference between the 'rules of grammar' and 'lexical entries' is in the generality, rather than the size, of the units to which they refer.

2.6 Overriding

In a default inheritance system, information is inherited only by default, i.e. in the absence of some exceptional information. One key question is how exceptions should be handled, and our answer is perhaps the most controversial part of this paper.

The standard answer is, of course, that any more general proposition is overridden by a more specific one that contradicts it. For example, the past-tense form *did* takes precedence over the expected **doed* because the former is specified in relation to DO, whereas the latter is inherited from the general rules for verbs. This principle, which we call **automatic overriding,** underlies most discussions of inheritance (e.g. Shieber 1986; Flickinger 1987), but it is also assumed in a lot of linguistic theory where the notion of 'inheritance' is not recognized as such—e.g. in the 'Proper Inclusion Precedence Principle' governing the ordering of rules in phonology (see, for example, Pullum 1979 for a survey of this literature).

Our answer is quite different, and involves the negative propositions, introduced by 'NOT:', which we described earlier. In WG, inheritance is not blocked by a more specific proposition, but by a negative proposition. We know that **doed* is not possible because there is a proposition that tells us so (24a), and not just because (24b) requires *did:*

(24a) NOT: whole of past DO = stem of it + whole of mEd.

(24b) whole of past DO = <did>.

Every exceptional fact is paired with a negative fact that blocks inheritance. We call this **stipulated overriding.** It remains to be seen which of these approaches—automatic overriding or stipulated overriding—will be favored by future research in NLP. The extra cost of exceptional facts in the first system lies in the need to ensure that more specific facts are accessed before more general ones. In the second system, the cost lies in the need for a larger database. Our reasons for preferring stipulated overriding are partly concerned with cognitive modeling (see Hudson 1990: 40ff), but we also believe that the syntactic and semantic arguments that we present in the next sections support this approach. Here, then, is the rule of inference for default inheritance:

Default inheritance
If A isa B, then for any true proposition P that refers to B, it is possible to infer another true proposition Q that is the same as P except that A is substituted in Q for B in P, **unless NOT: Q.**

That is, we can apply the inheritance rule defined in the last section, unless there is a negative proposition that conflicts with the inherited proposition. This negative proposition has the form NOT: Q, where Q is also a proposition that is available either by inspection or by inference. This proposition, NOT: Q, must itself pass the same test, since it may in turn be overridden by NOT: NOT: Q, and so on recursively.

We can now give a more detailed summary of inheritance and blocking in WG.

(26a) A proposition P is **valid** iff

a. it is contained in the knowledge base or
bi. it may be inherited and
bii. NOT: P cannot be inherited.

(26b) A proposition P may be **inherited** iff

a. Q is valid and
b. at every point where P differs from Q, by containing Y instead of X, X subsumes Y.

(26c) A name X **subsumes** another name Y iff

a. Y isa X, or
b. Y is a compound name (A B), where B is subsumed by X, or
c. X = Y.

(Allowing inheritance to apply to compound names allows multiple inheritance—i.e., one concept may inherit down more than one path. For example, *dogs* is 'plural DOG,' an example of both DOG and 'plural noun.' From DOG it inherits its stem and its sense (inter alia), while 'plural noun' provides the suffix, the 'set' meaning, and the ability to occur, for example, after *these*).

Having introduced our theory of default inheritance, we can now discuss some linguistic applications in more depth. One of the most distinctive features of our theory is our claim that default inheritance applies to syntax and compositional semantics, so we shall concentrate on these areas. The preceding discussion should have made it clear that we also use default inheritance in morphology, but we will not pursue that further here. (A brief WG account of English inflectional morphology can be found in Hudson 1990: 181–90.)

3. Syntax

3.1 Word Types

WG syntax is centred on two inheritance hierarchies, one for word types (i.e. word classes and lexical items) and the other for grammatical relations. In Word Grammar

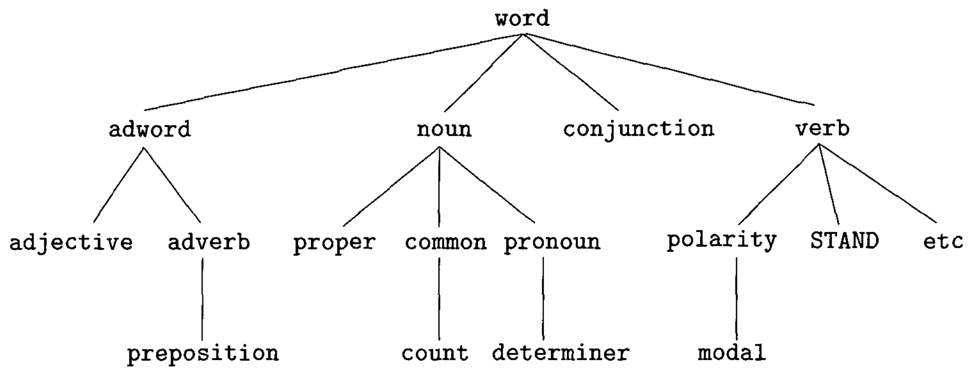


Figure 1
The word type hierarchy.

(as suggested by the name) the category 'word' is basic in every sense. Figure 1 shows the top of the hierarchy of word types assumed in WG for English, and some of the corresponding WG propositions are given in (27).

(27a) count isa common.

(27b) common isa noun.

(27c) noun isa word.

Three points should be noted about this hierarchy.

1. We assume a hierarchical relation among word types, instead of the more usual cross-classification based on features. This links to a general restriction on the use of features in WG, which excludes all features except those that are morphosyntactic—i.e., reflected in morphology and relevant to syntax or semantics.
2. Secondly, we assume some nonstandard analyses; in particular, a preposition is a kind of adverb, and a determiner is a kind of pronoun, which in turn is a kind of noun.
3. We prefer to keep an open mind on the extent to which our categories are universal, but we are sure that some are parochial (relevant to English only). This hierarchy can be continued downward to include lexical items (such as STAND, shown in the diagram), which may in turn be further subdivided; e.g., we can distinguish transitive and intransitive versions of STAND (with, it should be noted, the same irregular morphology in both cases):

(28a) STAND isa verb.

(28b) STAND/intrans isa STAND.

(28c) STAND/trans isa STAND.

As explained earlier, because lexical items are part of the same hierarchy as general word classes, there is no formal distinction between the lexicon and the rest of the grammar. Furthermore, we use the same isa relation to link word tokens to word types; so if w3 is the name of the word *stand* in *I can't stand cats*, it too will fit into the same hierarchy:

(29) w3 isa STAND/trans.

Word tokens can be thought of as a constantly changing fringe on the bottom of the (relatively) permanent hierarchy.

3.2 Grammatical Functions

We now come to the second hierarchy of syntax, the hierarchy of grammatical relations. Unlike most other syntactic theories, WG uses constituent structure only for the purpose of describing coordinate constructions (cf. Hudson 1990: 404ff for details). All other syntactic structure is expressed in terms of dependencies between pairs of words, one of which is the head of the other, its dependent. Higher nodes such as phrases or sentences are not represented explicitly in the grammar. WG is thus a variety of dependency grammar.

Dependency grammar was first formalized by Tesnière (1959) and refined by Hays (1964), Gaifman (1965), Robinson (1970) and others. A number of dependency-based theories have emerged from the linguistic underground during the last thirty years. These include the Meaning-Text model (Mel'čuk and Zolkovskij 1970; Mel'čuk 1988), Case Grammar (Anderson 1971; 1977), Daughter Dependency Grammar (Hudson 1976), WG (Hudson 1984; 1990), Functional Generative Description (Sgall, Hajiĉová, and Panevová 1986), and Lexicase (Starosta 1988). While none of these theories has attained widespread popularity, some of their central insights have become increasingly influential in the phrase structure grammar mainstream. For example, the trend toward head-driven approaches, the prominence of notions such as 'government,' the explicit use of grammatical relations and case, and the reduced amount of information carried in phrasal categories all reflect the general migration toward dependency. Increased interest in categorial grammars, and especially unification categorial grammars (which are virtually indistinguishable from dependency grammars) provides further evidence of this tendency.[3]

The combination of default inheritance with dependency syntax allows an interesting range of generalizations and exceptions. Like other dependency grammars, WG requires a typical word to have one head, though the same word may act as head to more than one other word, its dependents. As in other theories, just one word is allowed to be an exception to this rule; we call this word the 'root' of the sentence. This has (by definition) no head, and is generally a finite verb; e.g. in *Mary didn't jump*, the polarity verb ('auxiliary verb') *didn't* is the root, on which both *Mary* and *jump*

3 The last decade has seen increased interest in dependency grammar among computational linguists. Dependency grammar has been applied in the experimental parsing systems of Hellwig (1986), Sigurd (1989), and Covington (1991); in the 'Kielikone' natural language interface of Jappinen, Lassila, and Lahtola (1988); in the machine translation systems of EUROTRA (Johnson, King, and des Tombe 1985), DLT (Schubert 1987), Charles University (Sgall and Panevová 1987), and IBM Tokyo (Maruyama 1990); and in the speech recognition system of Giachin and Rullent (1989). Parsers based on the theories of Lexicase (Starosta and Nomura 1986) and Word Grammar (Fraser 1989; Hudson 1989) have also been developed.

depend. Here, then, we already have a simple example of default inheritance:

(30a) word has 1 head.

(30b) w1 isa word.

(30c) so: w1 has 1 head.

On the other hand, for w2, the finite verb *didn't*, this general rule is blocked to allow it to occur without a head (i.e. to make the head optional, '[0–1] head'). This analysis assumes that obligatory ('1') and optional ('[0–1]') conflict, so the former must be suppressed by (31d).[4]

(31a) finite verb has [0–1] head.

(31b) w2 isa finite verb.

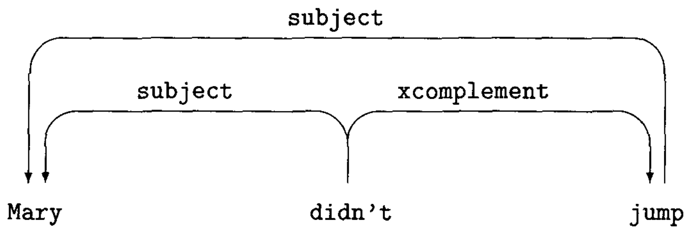
(31c) so: w2 has [0–1] head.

(31d) NOT: finite verb has 1 head.

(31e) so: NOT: w2 has 1 head.

If the rule about having one head per word allows exceptions in one direction, we may expect exceptions in the other direction as well: words that have more than one head. This is not allowed in other versions of dependency grammar,[5] but in WG it is the basis for our analysis of a range of important constructions: raising, control, extraction, and passives (not to mention coordination, which is often allowed as an exception by other theories). For example, in *Mary didn't jump*, we recognize *Mary* as the subject not only of *didn't* but also of *jump*, so *Mary* has two heads, contrary to the general rule.

(32)



4 This analysis may in fact be more complicated than it needs to be. We could allow finite verbs to inherit the regular '1 head' simply by not blocking it, and allow for '0 head' by an extra rule, which provides the other alternative.

5 The notion of a word with two heads is meaningless in theories based on phrase structure, because 'head' is used there in relation to phrases, not words. The X-bar 'head' corresponds to our 'root,' the word in a phrase that has no head inside that phrase. It is true that some linguists have suggested that a phrase might have more than one head (e.g. Warner 1987), and this has been a standard analysis of coordinate structures since Bloomfield (1933); but this is very different from a single word having more than one head.

(In a dependency diagram, the arrow points towards the dependent.) This is permitted by a proposition which, at least by implication, overrides the general rule, and which refers to the grammatical function 'xcomplement':[6]

(33) subject of xcomplement of word = subject of it.

In other words, a word may have two heads provided that one of them is the xcomplement of the other. (We return below to the relations among the grammatical functions such as 'subject' and 'xcomplement').

The possibility of having more than one head is related to another important generalization, namely that heads and dependents are usually adjacent. If we think of each word as defining a 'phrase,' made up of that word plus any words subordinate to it, this is equivalent to the PSG ban on discontinuous phrases. In the simple cases, then, the following generalization is true:

(34) position of word = adjacent-to head of it.

An operational definition of 'adjacent-to' checks that no word between the words concerned has a head outside the phrase:

(35a) A is **adjacent-to** B iff every word between A and B is a subordinate of B.

(35b) A is a **subordinate** of B iff A is B or A is a dependent of a subordinate of B.

But what if a word has more than one head? This normally leads to a discontinuity; e.g. in *Mary didn't jump*, the phrase rooted in *jump* consists of *Mary jump*, but does not include *didn't*. Saying that *Mary jump* is discontinuous is the same as saying that *Mary* is not adjacent to one of its heads, *jump*. Interestingly, though, *Mary* does have one head to which it is adjacent (viz *didn't*), and more generally the same is true of all discontinuities: even if a word has some nonadjacent heads, it also has at least one to which it is adjacent. We can therefore keep our generalization (34) in a slightly revised form, with 'a head' (one head) rather than 'head' (every head):

(36) position of word = adjacent-to a head of it.

This generalization is inherited by every word, so every word has to be adjacent to at least one of its heads. This treatment of discontinuity has many important ramifications that cannot be explored fully here.

The generalizations discussed in this section have referred crucially to grammatical functions.[7] In some cases these were the functions 'dependent' and 'head,' but we also mentioned 'subject' and 'xcomplement.' The functional categories are arranged in an

6 The name 'xcomplement' is borrowed from Lexical Functional Grammar. The term used in earlier WG literature is 'incomplement.'

7 As in LFG, the term 'function' is used here in both its mathematical and grammatical senses, but (unlike LFG) with a single word as the argument; so in expressions such as 'head of X' or 'subject of X,' X is always some word or word type rather than a phrase.

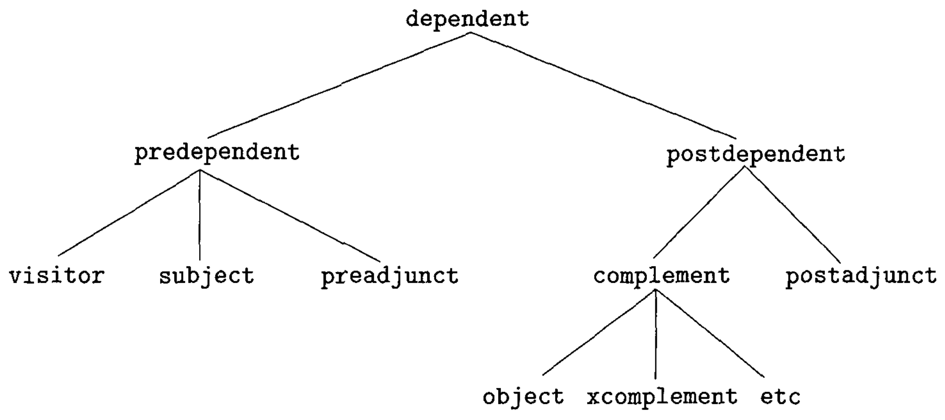


Figure 2
Hierarchy of dependency types for English.

inheritance hierarchy, and the one for English is shown (in part) in Figure 2. This hierarchy allows generalizations to be made about different types of dependent at the most appropriate level. As with the hierarchy of word classes, we are sure that some of these categories are specific to languages like English, and not universal, but others seem to be very widespread or universal.

Generalizations about word order are perhaps the clearest examples of generalizations that take advantage of the hierarchical organization of grammatical functions in WG. Proposition (37) states the default word order of English (i.e. English is a head-first language).

(37) position of dependent of word = after it.

Although this generalization has important exceptions, it is clearly true of 'typical' dependencies in English; for example, in a running text we find that between 60% and 70% of dependencies are head-first.

The exceptional order of those dependent types that typically precede their heads is handled by the propositions shown in (38), referring to the super-category 'predependent.'

(38a) position of predependent of word = before it.

(38b) NOT: position of predependent of word = after it.

The usual machinery of default inheritance applies, so that (38b) blocks the normal head-first rule, and (38a) replaces it by the exceptional one. There are just a few constructions that allow a dependent to precede its head, one of which is the subject-verb pair.[8]

8 As one of our readers commented, if pressure toward consistency were the strongest pressure on language development, we should expect VSO languages to outnumber SVO, but of course they do not (about 40% of the world's languages are said to be SVO, compared with only 10% VSO). One explanation for this is presumably the strong tendency for subjects to be more topical than verbs, but it remains as a challenging area for research.

One of the most important applications of default inheritance in WG syntax is in the distinction of 'derived' from 'underlying' or 'basic' patterns. The general point is that underlying patterns are allowed by the most general rules, and are therefore most typical; whereas derived patterns involve rules that override these, so they are exceptional. In this way we can capture the different statuses of these patterns in a completely monostratal analysis and without the use of special devices such as transformations, lexical rules, or metarules.

Take for instance the rules given in the Appendix for inverted subjects.

(39a) tensed polarity-verb has 1 sv-order.

(39b) sv-order of verb = {/: s+v, v+s}.

(39c) position of dependent of word = after it.

(39d) position of predependent of word = before it.

(39e) NOT: position of predependent of word = after it.

(39f) NOT: position of subject of v+s verb = before it.

(39g) NOT: NOT: position of subject of v+s verb = after it.

The first two rules allow us to distinguish tensed polarity-verbs according to whether their subject precedes ('s+v') or follows ('v+s') them.[9] This allows us to treat 'v+s verb' as an exception to the general rule that subjects precede their head, which is in turn an exception to the generalization that words follow their heads. This system allows us to generate a sentence such as *Did Mary jump?* with just one syntactic structure, free of empty positions, while still showing[10] that it is a less normal construction than a sentence such as *Mary did jump*. In parsing terms, the only problem is to find and apply the necessary propositions; there is no need to reconstruct any kind of abstract structure for the sentence itself.

The use of 'NOT' rules for overriding defaults finds support in the fact that the 'NOT' rule in (39e) is also crucial for solving at least two other major problems, namely passives and extraction. In a passive sentence like (40), WG handles object-promotion by analyzing the subject as also being the object. This is achieved by means of proposition (41a) (which is slightly simplified).

(40) Mary was kissed by John.

(41a) subject of passive verb = object of it.

(41b) NOT: position of predependent of word = after it.

The problem is that *Mary*, as the object of *kissed*, ought to follow it, but since *Mary* is also the subject this requirement is overridden by proposition (39e=41b), so *Mary* never inherits the need to follow *kissed*.[11]

9 See Hudson (1990: 215–6) for a discussion of the apparent lack of morphological consequences of the subject-position feature.

10 The exceptionality of an inverted subject is shown by the negative proposition 'NOT: NOT: position of subject of v+s verb = after it.' This proposition is inherited by the word tokens concerned—i.e. it is part of the analysis of the sentence itself, and not just available in the grammar.

11 It may not be obvious exactly how this works. How does (41b) stop the object of a passive from following the verb, given that it refers to 'predependent,' which does not subsume 'object'? The answer lies in (41a): the object of a passive verb is also its subject, so any rule (such as (41b)) that applies to the subject also, ipso facto, applies to its object.

A similar approach is used to handle extraction. For example, consider sentence (42).

(42) Salesmen I distrust.

Here *salesmen* must be an object of *distrust,* so the order should be *I distrust salesmen,* but in this case we also recognize a special kind of predependent relation ('visitor,' roughly equivalent to 'Comp') between *distrust* and *salesmen,* so once again the word order conflict between these two relations can be resolved. The rules are given in (43).

(43a) finite verb has [0–1] visitor.

(43b) visitor isa predependent.

(43c) position of predependent of word = before it.

(43d) NOT: position of predependent of word = after it.

(43e) visitor of word = a postdependent of it.

(43f) visitor of word = a visitor of complement of it.

Proposition (43a) allows *distrust* to have a visitor, which according to (43b) is a kind of predependent and therefore, by inheritance from (43c,d), must precede it. The visitor is also some kind of postdependent, according to (43e), so it may be the verb's object as in our example. But equally it may 'hop' down the dependency chain thanks to (43f), thereby providing for the analysis of sentences such as (44).

(44) Salesmen I don't think many people say they trust.

Further details of the WG analysis of passives and extraction can be found in Hudson (1990).

Both passivization and extraction are standard examples of syntactic problems that need special machinery. According to WG—and more recently Flickinger, Pollard, and Wasow (1985) and Flickinger (1987)—all that is needed is default inheritance, which is available (though generally not explicitly recognized) in every linguistic theory; so any theory capable of accommodating exceptional morphology already has the power to deal with subject-inversion, passives and extraction.

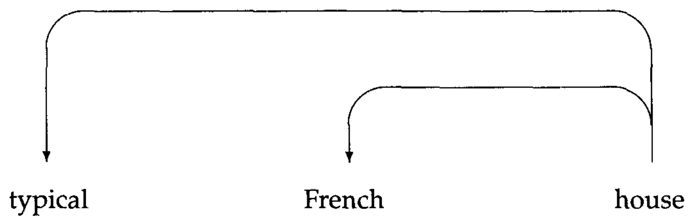
4. Semantics

Default inheritance also plays a crucial part in the WG treatment of semantics. It is probably obvious how it applies in the familiar examples of inheritance in semantic networks—e.g., how one can infer that Clyde has a head from more general propositions about elephants or animals. Rather than discussing this familiar territory we shall show how default inheritance helps us to answer a recurrent objection to dependency analysis: the syntactic dependency structure is completely flat, so it does not provide any units between the individual word and the complete phrase (where 'phrase' means a word and the complete set of all its dependents and their respective phrases).

For example, the semantic structure for the phrase *typical French house* has to mention the concept 'French house' (a typical French house is a house that is typical as a

French house, and not just as a house); but a flat dependency structure such as (45) provides no syntactic unit larger than the individual words (Dahl 1980).

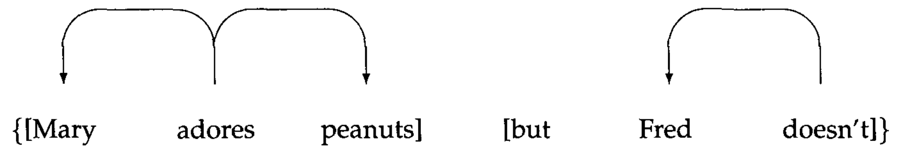
(45)



Similarly, how can we handle 'VP-anaphora' without a VP node? For example, we need the concept 'adore peanuts' as part of the semantic structure of *Fred doesn't* in (46a), but *adores peanuts* is not a phrase in the syntactic structure of the first clause:

(46a) Mary adores peanuts but Fred doesn't.

(46b)



Inheritance is relevant to these questions because notions such as 'French-house'[12] and 'adoring-peanuts' can be located in an inheritance hierarchy between the more general notions denoted by their heads ('house' or 'adoring') and the more specific ones denoted by the complete phrase ('typical-French-house,' 'Mary-adoring-peanuts'):

(47)

$$\begin{tikzcd}[no arrows] \text{house} \arrow[d, dash] & \text{adoring} \arrow[d, dash] \\ \text{French-house} \arrow[d, dash] & \text{adoring-peanuts} \arrow[d, dash] \\ \text{typical-French-house} & \text{Mary-adoring-peanuts} \end{tikzcd}$$

As usual, each of these concepts inherits all the properties of the concepts above it in the hierarchy except where these are overridden.[13]

It is easy to see how a dependency grammar can generate concepts at the top and bottom of these hierarchies. The top concept comes straight from the lexical entry for the root word (e.g. HOUSE, ADORE), and the bottom one belongs to the word token—the word in the sentence concerned (e.g. the word *house* in the phrase *typical*

12 The hyphen in 'French-house' is needed because this is an atomic name whose internal structure is irrelevant.

13 Overriding is found in well-known examples such as *fake diamond,* not to mention ordinary negative sentences such as *Mary didn't jump.* A fake diamond is an object that inherits some of the properties of diamonds, especially the visible ones, but not all, and in particular not those that are criterial in the trade. If the sense of *Mary jumped* (i.e. the kind of situation to which it can refer) is P, then the referent of *Mary didn't jump* (i.e. the actual situation to which it refers) is NOT: P, in which we know nothing about the situation except that it is not one in which Mary jumped. (The relevant rule is given in the Appendix).

French house). The problem is how to generate concepts in between the two, and the WG solution is to recognize the notion **head-sense**: the head-sense of some word W is the concept that results from combining W with its head. Thus the head-sense of *French* in our example is the result of combining *French* with *house* (as adjunct and head respectively); and that of *peanuts* in *Mary adores peanuts* is the result of combining *peanuts* with *adores*. This is how we generate semantic structures that contain 'French-house' and 'adoring-peanuts' without recognizing *French house* or *adores peanuts* as units in the syntax.

The rules that allow head-senses include these:

(48a) dependent of word has 1 head-sense.

(48b) referent of word = a dependent of head-sense of it.

By the first rule, every dependent of a word has a head-sense, i.e. makes a distinct contribution, in combination with its head, to the sentence's meaning. The notion 'head-sense' is thus a functor that maps the senses of the dependent and the head onto a third concept, applying equally to complements such as *peanuts* in *adores peanuts* and to adjuncts such as *French* in *French houses*. There is nothing quite like it in standard semantic systems such as Montague Grammar, but it applies in conjunction with rather more standard functors which each pick out one particular (semantic) role as the one filled by the dependent's referent (by rule (48b)). Generally speaking, this role is defined by just one of the words concerned, according to whether the dependent is an adjunct or a complement. If it is an adjunct, it defines its own semantic role (e.g. *French* defines its own semantic role as 'nationality' or 'location'), but if it is a complement then it leaves the head to define its role (e.g. *adores* provides the role 'adoree,' or whatever it should be called, for its complement).

The relevance to inheritance is that all the different head-senses are held together by inheritance. These are the basic rules:

(49a) head-sense of dependent of word isa sense of it.

(49b) referent of word isa head-sense of dependent of it.

That is, the combination of a word and one of its dependents yields a concept that is normally a particular case of the concept defined by the word on its own (a French house is a kind of house; adoring peanuts is a kind of adoring), and whatever a word refers to must be a particular case of all the concepts defined by it in combination with its dependents (e.g., the particular situation referred to by *Mary adores peanuts* must be one that is both an example of adoring peanuts and of Mary adoring).

We can impose further structure on the semantics; for example, by requiring all other head-senses to build on that of the object:

(50) head-sense of dependent of word isa head-sense of object of it.

This is equivalent to an ordered procedure that combines the verb with its object before it takes account of any other dependents. This has the attraction of a Categorial Grammar approach, in which dependents are added one at a time and subtle distinctions of grouping can be made e.g., among the complements within VP; but it has the advantage of not requiring a similar binary bracketing in the syntax. Why is this an advantage? Some of the drawbacks of binary syntactic structures are obvious—e.g., the need for far more phrasal nodes each carrying much the same information. However, the advantage of our system that we should like to draw attention to here is the possibility of reducing the amount of ambiguity.

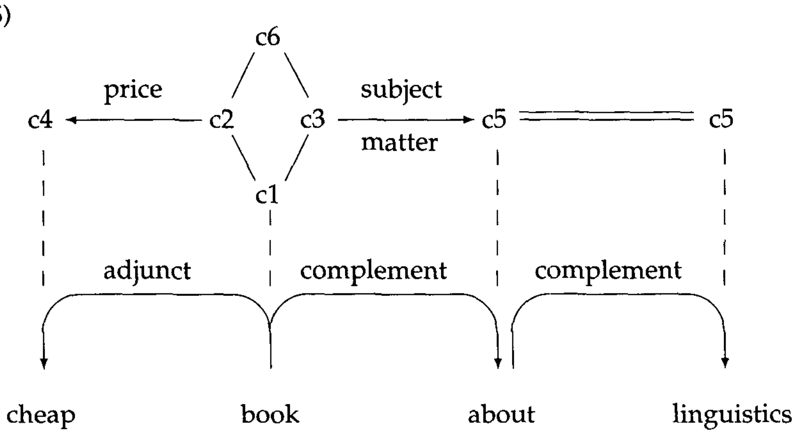
For example, in WG the sequence *cheap book about linguistics* has just one possible syntactic analysis, the usual flat analysis with both *cheap* and *about* as dependents of *book*, but its semantics contains the concepts 'cheap book' and 'book about linguistics' respectively. These interpretations can be seen from examples such as the following, where the sense of ONE is based on the semantic structure of the antecedent phrase.

(51a) I wanted a cheap book about linguistics but I could only find one about cricket.

(51b) I wanted a cheap book about linguistics but I could only find a dear one.

In a standard approach, the first clause has to be given two distinct syntactic structures, one containing the unit *cheap book* and the other *book about linguistics*; but this ambiguity cannot be resolved until the end of the second clause. Our judgment is that the first clause is not in fact semantically ambiguous in this way; and according to our approach there is no need to postulate such ambiguity since both concepts 'cheap book' and 'book about linguistics' are available there, in addition to the unifying concept 'cheap book about linguistics' (which could be identified in relation to *book* as its **supersense**, the concept that is an instance of the head-sense of every dependent). Here is the relevant part of the structure of both (51a) and (51b):

(55)



In this diagram, the concepts are as follows:

c1	'book'
c2	'cheap book'
c3	'book about linguistics'
c4	'cheap'
c5	'about linguistics'
c6	'cheap book about linguistics'

The isa relations in this structure provide the basis for ordinary default inheritance; so if a book has pages, then so do a cheap book, a book about linguistics, and a cheap book about linguistics. They also allow defaults to be overridden in the usual way; this is precisely the function of ordinary adjectives and adverbs like CHEAP or QUICKLY, which mean 'more cheap/quickly than the default,' where the default value is the average for the category concerned. The theoretical apparatus that allows these meanings is precisely the same as the one we apply in morphology and syntax.

5. Conclusion

We have shown that it is possible, and fruitful, to develop a general theory of default inheritance that is equally relevant, and equally important, for all levels of linguistic analysis. We have demonstrated this in relation to syntax and semantics, and by occasional allusions to morphology, but we assume it is equally true of phonology and of pragmatics (in every sense).

This conclusion, if true, is important for linguists and psychologists, because it shows that the structures found within language are formally similar in at least some respects to those found in general knowledge, and that both kinds of knowledge are processed in ways that are similar in at least some important respects. And our conclusion is also important for computational linguists because it indicates the possibility of a single very general inheritance mechanism that can be applied at every stage in the parsing process, and also in the manipulation of knowledge structures. A second conclusion, however, is that default inheritance should be based on the principle of stipulated overriding (by means of 'NOT:...' propositions), rather than on automatic overriding. This conclusion conflicts directly with standard theory and practice.

References

Anderson, John M. (1971). *The Grammar of Case: Towards a Localistic Theory.* Cambridge: Cambridge University Press.

Anderson, John M. (1977). *On Case Grammar: Prolegomena to a Theory of Grammatical Relations.* London: Croom Helm.

Bloomfield, Leonard (1933). *Language.* London: Allen and Unwin.

Brachman, Ronald J. (1985). "I lied about the trees, or defaults and definitions in knowledge representation." *AI Magazine* 6, 80–93.

Calder, Jonathan (1989). "Paradigmatic morphology." In *Proceedings, Fourth Conference of the European Chapter of the Association for Computational Linguistics,* Manchester, U.K., April, 58–59.

Cook, S., ed. (1989). *Proceedings, 1989 European Conference on Object Oriented Programming.* Cambridge: Cambridge University Press.

Covington, Michael A. (1991). "Parsing discontinuous constituents in dependency grammar." *Computational Linguistics* 16(4), 234–236.

Daelemans, Walter (1987). *Studies in language technology: An object-oriented computer model of morphosyntactic aspects of Dutch.* Doctoral dissertation, Katholieke Universiteit Leuven.

Dahl, Östen (1980). "Some arguments for higher nodes in syntax: A reply to Hudson's 'Constituency and dependency'." *Linguistics* 18, 485–488.

De Smedt, Koenraad (1984). "Using object-oriented knowledge representation techniques in morphology and syntax programming." *ECAI-84,* 181–184.

Etherington, David R., and Reiter, Raymond (1983). "On inheritance hierarchies with exceptions." *Proceedings, AAAI-83.* Washington, 104–108.

Etherington, David R. (1988). *Reasoning with Incomplete Information.* Los Altos: Morgan Kaufmann.

Flickinger, Daniel P. (1987). *Lexical rules in the hierarchical lexicon.* Doctoral dissertation, Stanford University, California.

Flickinger, Daniel P.; Pollard, Carl J.; and Wasow, Thomas (1985). "Structure sharing in lexical representation." In *Proceedings, 23rd Annual Meeting of the Association for Computational Linguistics,* Chicago, 262–267.

Fraser, Norman M. (1989). "Parsing and dependency grammar." In *UCL Working Papers in Linguistics* 1, edited by Robyn Carston, 296–319. London: University College London.

Gaifman, Haim (1965). "Dependency systems and phrase-structure systems." *Information and Control* 8, 304–337.

Giachin, Egidio P., and Rullent, Claudio (1989). "A parallel parser for spoken natural language." *IJCAI-89,* 1537–1542.

Gibbon, Dafydd (1990). "Prosodic association by template inheritance." In *Proceedings, Workshop on Inheritance in Natural Language Processing*, edited by Walter Daelemans and Gerald Gazdar, 65–81. Tilburg: ITK.

Hays, David (1964). "Dependency theory: a formalism and some observations." *Language* 40, 511–525.

Hellwig, Peter (1986). "Dependency Unification Grammar (DUG)." *COLING-86*, 195–198.

Hudson, Richard A. (1976). *Arguments for a Non-Transformational Grammar.* Chicago: University of Chicago Press.

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

Hudson, Richard A. (1989). "Towards a computer testable word grammar of English." In *UCL Working Papers in Linguistics* 1, edited by Robyn Carston, 321–339. London: University College London.

Hudson, Richard A. (1990). *English Word Grammar.* Oxford: Blackwell.

Jackendoff, Ray (1983). *Semantics and Cognition.* Cambridge, MA: The MIT Press.

Jappinen, Harri; Lassila, Eero; and Lehtola, Aarno (1988). "Locally governed trees and dependency parsing." *COLING-88*, 275–277.

Johnson, Rod; King, Maghi; and des Tombe, Louis (1985). "EUROTRA: A multilingual system under development." *Computational Linguistics* 11, 155–169.

Maruyama, Hiroshi (1990). "Structural disambiguation with constraint propagation." In *Proceedings, 28th Annual Meeting of the Association for Computational Linguistics*, 31–38.

Mel'čuk, Igor A. (1988). *Dependency Syntax: Theory and Practice.* Albany, NY: State University of New York Press.

Mel'čuk, Igor A., and Zolkovskij, Alexander K. (1970). "Towards a functioning 'Meaning-Text' model of language." *Linguistics* 57, 10–47.

Pullum, Geoffrey K. (1979). *Rule Interaction and the Organization of a Grammar.* London: Garland.

Reinhard, Sabine, and Gibbon, Dafydd (1991). "Prosodic inheritance and morphological generalisations." In *Proceedings, Fifth Conference of the European Chapter of the Association for Computational Linguistics.* April, Berlin, 131–136.

Robinson, Jane J. (1970). "Dependency structures and transformational rules." *Language* 46, 259–285.

Schubert, Klaus (1987). *Metataxis: Contrastive Dependency Syntax for Machine Translation.* Dordrecht: Foris.

Sgall, Petr; Hajičová, Eva; and Panevová, Jarmila (1986). *The Meaning of the Sentence in its Semantic and Pragmatic Aspects.* Prague: Academia.

Sgall, Petr, and Panevová, Jarmila (1987). "Machine translation, linguistics, and interlingua." In *Proceedings, Third Conference of the European Chapter of the Association for Computational Linguistics*, 99–108.

Shieber, Stuart M. (1986). *An Introduction to Unification-Based Approaches to Grammar.* Stanford, CA: CSLI.

Sigurd, Bengt (1989). "DEPARSE—An experimental dependency parser." In *Praktisk Lingvistik* 12, edited by B. Sigurd; M. Eeg-Olofsson; L. Eriksson; B. Gawronska-Werngren; K. Holmqvist; and P. Touati, 30–41. Lunds: Lunds Universitet.

Starosta, Stanley (1988). *The Case for Lexicase: An Outline of Lexicase Grammatical Theory.* London: Pinter.

Starosta, Stanley, and Nomura, H. (1986). "Lexicase parsing: A lexicon-driven approach to syntactic analysis." *COLING-86*, 127–132.

Tesnière, Lucien (1959). *Éléments de Syntaxe Structurale.* Paris: Librairie Klincksieck.

Thomason, Richmond H., and Touretzky, David F. (1991). "Inheritance theory and networks with roles." In *Principles of Semantic Networks*, edited by John F. Sowa, 231–266. San Mateo, CA: Morgan Kaufmann.

Touretzky, David F. (1986). *The Mathematics of Inheritance Systems.* Los Altos: Morgan Kaufmann.

Warner, Anthony (1987). "Multiple heads and minor categories in GPSG." *Linguistics* 27, 179–205.

Appendix

This appendix contains a selection of WG propositions that are sufficient to generate simple syntactic, morphological, and semantic structures for the sentences *Mary jumped, Mary didn't jump, Did Mary jump?,* and *Didn't Mary jump?* After presenting the grammar, we give an analysis of *Mary jumped.*

GRAMMAR

Word classes
noun isa word.
proper isa noun.
verb isa word.
polarity-verb isa verb.

Morphosyntactic features
verb has 1 finiteness.
finiteness of verb = {/: finite, non-finite}.
non-finite verb has 1 aspect.
aspect of verb = {/: infinitive, perfect, participle}.
finite verb has 1 mood.
mood of verb = {/: tensed, imperative}.
tensed verb has 1 tense.
tense of verb = {/: past, present}.
tensed polarity-verb has 1 polarity.
polarity of verb = {/: positive, negative}.
tensed polarity-verb has 1 sv-order.
sv-order of verb = {/: s+v, v+s}.

Regular morphology
body isa form.
ed-form isa body.
base isa body.
n't-form isa form.
past verb has 1 ed-form.
whole of ed-form of verb = stem of it + whole of mEd.
whole of mEd = <ed>.
negative verb has 1 n't-form.
whole of n't-form of verb = whole of body of it + whole of mN't.
whole of mN't = <n't>.
infinitive verb has 1 base.
whole of base of word = stem of it.

Dependencies
predependent isa dependent.
postdependent isa dependent.
subject isa predependent.
complement isa postdependent.
xcomplement isa complement.

Syntactic valency
word has 1 head.
finite verb has [0-1] head.
word has 0 subject.
verb has 1 subject.
NOT: tensed verb has 0 subject.
subject of verb isa noun.
word has 0 complement.
polarity-verb has 1 xcomplement.
xcomplement of polarity-verb isa infinitive verb.

Raising/control
subject of xcomplement of word = subject of it.

Word order
position of dependent of word = after it.
position of predependent of word = before it.
NOT: position of predependent of word = after it.
NOT: position of subject of v+s verb = before it.
NOT: NOT: position of subject of v+s verb = after it.

Semantics: sense and reference
word has 1 meaning.
referent isa meaning.
sense isa meaning.
proper has 0 sense.
NOT: proper has 1 sense.
referent of word isa sense of it.
NOT: referent of negative verb isa sense of it.
NOT: proper has 1 sense.

Semantics: words and other actions
word isa action.
action has 1 time.
action has 1 actor.
actor of sense of word = referent of subject of it.
time of referent of past verb = before it.

'Lexicon'
MARY isa proper.
whole of MARY = <Mary>.
referent of MARY = Mary.

DO isa verb.
DO/dummy isa DO.
DO/dummy isa polarity-verb.
stem of DO = <do>.
whole of past DO = <did>.
NOT: whole of past DO = stem of it + whole of mEd.
meaning of DO/dummy = meaning of xcomplement of it.

JUMP isa verb.
stem of JUMP = <jump>.
sense of JUMP = jumping.
jumping isa action.

SENTENCE ANALYSIS

The following is an analysis of *Mary jumped,* in which we present all the propositions about its constituent words (w1, w2) that can be inherited from the grammar on the basis of the propositions in B, the 'calculated' properties. The latter are of course not generated by inheritance but by applying some kind of 'Best Fit Principle' to the observable propositions in A plus the already inheritable propositions.

A. Observable propositions
whole of w1 = <Mary>.
whole of w2 = <jumped>.
position of w1 = before w2.

B. Calculated propositions
w1 isa singular MARY.
w2 isa past tensed finite JUMP.
subject of w2 = w1.
head of w1 = w2.
finiteness of w2 = finite.
mood of w2 = tensed.
tense of w2 = past.
sense of w2 = c1.
referent of w2 = c2.

C. Inherited propositions
w1 isa proper.
w1 isa noun.
w1 isa word.
w1 isa action.
w1 has 1 time.
w1 has 1 actor.
w1 has 1 head.
w1 has 1 referent.
referent of w1 = Mary.

w2 isa past tensed finite verb.
w2 isa tensed finite verb.
etc.
w2 isa verb.
w2 has 1 finiteness.
w2 has 1 mood.
w2 has 1 tense.
w2 has [0-1] head.
w2 has 1 subject.
subject of w2 isa noun.

```
position of subject of w2 = before w2.
w2 has 0 complement.
w2 has 1 sense.
w2 has 1 referent.

c1 = jumping.
c1 isa action.
c1 has 1 time.
c1 has 1 actor.
c2 isa jumping.
c2 isa action.
c2 has 1 time.
c2 has 1 actor.
time of c2 = before w2.

actor of c1 = Mary.
actor of c2 = Mary.
```