PARSING HEAD-DRIVEN PHRASE STRUCTURE GRAMMAR

Derek Proudian and Carl Pollard
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA. 94303, USA

Abstract

The Head-driven Phrase Structure Grammar project (HPSG) is an English language database query system under development at Hewlett-Packard Laboratories. Unlike other product-oriented efforts in the natural language understanding field, the HPSG system was designed and implemented by linguists on the basis of recent theoretical developments. But, unlike other implementations of linguistic theories, this system is not a toy, as it deals with a variety of practical problems not covered in the theoretical literature. We believe that this makes the HPSG system unique in its combination of linguistic theory and practical application.

The HPSG system differs from its predecessor GPSG, reported on at the 1982 ACL meeting (Gawron et al. [1982]), in four significant respects: syntax, lexical representation, parsing, and semantics. The paper focuses on parsing issues, but also gives a synopsis of the underlying syntactic formalism.

1 Syntax

HPSG is a lexically based theory of phrase structure, so called because of the central role played by grammatical heads and their associated complements.* Roughly speaking, heads are linguistic forms (words and phrases) that exert syntactic and semantic restrictions on the phrases, called complements, that characteristically combine with them to form larger phrases. Verbs are the heads of verb phrases (and sentences), nouns are the heads of noun phrases, and so forth.

As in most current syntactic theories, categories are represented as complexes of feature specifications. But the HPSG treatment of lexical subcategorization obviates the need in the theory of categories for the notion of bar-level (in the sense of X-bar theory, prevalent in much current linguistic research). In addition, the augmentation of the system of categories with stack-valued features - features whose values are sequences of categories - unifies the theory of lexical subcategorization with the theory of binding phenomena. By binding phenomena we mean essentially non-clause-bounded dependencies, such as those involving dislocated constituents, relative and interrogative pronouns, and reflexive and reciprocal pronouns [12].

More precisely, the subcategorization of a head is encoded as the value of a stack-valued feature called "SUBCAT". For example, the SUBCAT value of the verb *persuade* is the sequence of three categories [VP, NP, NP], corresponding to the grammatical relations (GR's): controlled complement, direct object, and subject respectively. We are adopting a modified version of Dowty's [1982] terminology for GR's, where *subject* is last, *direct object* second-to-last, etc. For semantic reasons we call the GR following a controlled complement the *controller*.

One of the key differences between HPSG and its predecesor GPSG is the massive relocation of linguistic information from phrase structure rules into the lexicon [5]. This wholesale lexicalization of linguistic information in HPSG results in a drastic reduction in the number of phrase structure rules. Since rules no longer handle subcategorization, their sole remaining function is to encode a small number of language-specific principles for projecting from lexical entries to surface constituent order.

The schematic nature of the grammar rules allows the system to parse a large fragment of English with only a small number of rules (the system currently uses sixteen), since each rule can be used in many different situations. The constituents of each rule are sparsely annotated with features, but are fleshed out when taken together with constituents looked for and constituents found.

For example the sentence *The manager works* can be parsed using the single rule R1 below. The rule is applied to build the noun phrase *The manager* by identifying the head *H* with the lexical element *manager* and the complement *C1* with the lexical element *the*. The entire sentence is built by identifying the *H* with *works* and the *C1* with the noun phrase described above. Thus the single rule R1 functions as both the $S \rightarrow NP\ VP$, and $NP \rightarrow Det\ N$ rules of familiar context free grammars.

```
R1.  x  ->  c1  h[(CONTROL INTRANS)]  a*
```

Figure 1. A Grammar Rule.

* HPSG is a refinement and extension of the closely related Generalized Phrase Structure Grammar [7]. The details of the theory of HPSG are set forth in [11].

Feature Passing

The theory of HPSG embodies a number of substantive hypotheses about universal grammatical principles. Such principles as the Head Feature Principle, the Binding Inheritance Principle, and the Control Agreement Principle, require that certain syntactic features specified on daughters in syntactic trees are inherited by the mothers. Highly abstract phrase structure rules thus give rise to fully specified grammatical structures in a recursive process driven by syntactic information encoded on lexical heads. Thus HPSG, unlike similar "unification-based" syntactic theories, embodies a strong hypothesis about the flow of relevant information in the derivation of complex structures.

Unification

Another important difference between HPSG and other unification based syntactic theories concerns the form of the expressions which are actually unified. In HPSG, the structures which get unified are (with limited exceptions to be discussed below) not general graph structures as in Lexical Functional Grammar [1], or Functional Unification Grammar [10], but rather flat atomic valued feature matrices, such as those shown below.

```
[(CONTROL O INTRANS) (MAJ N A)
 (AGR 3RDSG) (PRD MINUS) (TOP MINUS)]

[(CONTROL O) (MAJ N V) (INV PLUS)]
```

Figure 2. Two feature matrices.

In the implementation of HPSG we have been able to use this restriction on the form of feature matrices to good advantage. Since for any given version of the system the range of atomic features and feature values is fixed, we are able to represent flat feature matrices, such as the ones above, as vectors of integers, where each cell in the vector represents a feature, and the integer in each cell represents a disjunction of the possible values for that feature.

CON	MAJ	AGR	PRD	INV	TOP	...
17	10	2	1	3	1	
1	12	7	3	1	3	

Figure 3: Two transduced feature matrices.

For example, if the possible values of the *MAJ* feature are N, V, A, and P then we can uniquely represent any combination of these features with an integer in the range 0..15. This is accomplished simply by assigning each possible value an index which is an integral power of 2 in this range and then adding up the indices so derived for each disjunction of values encountered. Unification in such cases is thus reduced to the "logical and" of the integers in each cell of the vector representing the feature matrix. In this way unification of these flat structures can be done in constant time, and since "logical and" is generally a single machine instruction the overhead is very low.

N	V	A	P	
1	0	1	0	= 10 = (MAJ N A)
1	1	0	0	= 12 = (MAJ N V)
				Unification
1	0	0	0	= 8 = (MAJ N)

Figure 4: Closeup of the MAJ feature.

There are, however, certain cases when the values of features are not atomic, but are instead themselves feature matrices. The unification of such structures could, in theory, involve arbitrary recursion on the general unification algorithm, and it would seem that we had not progressed very far from the problem of unifying general graph structures. Happily, the features for which this property of embedding holds, constitute a small finite set (basically the so called "binding features"). Thus we are able to segregate such features from the rest, and recurse only when such a "category valued" feature is present. In practice, therefore, the time performance of the general unification algorithm is very good, essentially the same as that of the flat structure unification algorithm described above.

2 Parsing

As in the earlier GPSG system, the primary job of the parser in the HPSG system is to produce a semantics for the input sentence. This is done compositionally as the phrase structure is built, and uses only locally available information. Thus every constituent which is built syntactically has a corresponding semantics built for it at the same time, using only information available in the phrasal subtree which it immediately dominates. This locality constraint in computing the semantics for constituents is an essential characteristic of HPSG. For a more complete description of the semantic treatment used in the HPSG system see Creary and Pollard [2].

Head-driven Active Chart Parser

A crucial difference between the HPSG system and its predecessor GPSG is the importance placed on the head constituent in HPSG. In HPSG it is the head constituent of a rule which carries the subcategorization information needed to build the other constituents of

the rule. Thus parsing proceeds head first through the phrase structure of a sentence, rather than left to right through the sentence string.

The parser itself is a variation of an active chart parser [4,9,8,13], modified to permit the construction of constituents head first, instead of in left-to-right order. In order to successfully parse "head first", an edge* must be augmented to include information about its span (i.e. its position in the string). This is necessary because head can appear as a middle constituent of a rule with other constituents (e.g. complements or adjuncts) on either side. Thus it is not possible to record all the requisite boundary information simply by moving a dot through the rule (as in Earley), or by keeping track of just those constituents which remain to be built (as in Winograd). An example should make this clear.

Suppose as before we are confronted with the task of parsing the sentence *The manager works*, and again we have available the grammar rule R1. Since we are parsing in a "head first" manner we must match the *H* constituent against some substring of the sentence. But which substring? In more conventional chart parsing algorithms which proceed left to right this is not a serious problem, since we are always guaranteed to have an anchor to the left. We simply try building the leftmost constituent of the rule starting at the leftmost position of the string, and if this succeeds we try to build the next leftmost constituent starting at one position to the right of wherever the previous constituent ended. However in our case we cannot assume any such anchoring to the left, since as the example illustrates, the *H* is not always leftmost.

The solution we have adopted in the HPSG system is to annotate each edge with information about the span of substring which it covers. In the example below the inactive edge E1 is matched against the head of rule R1, and since they unify the new active edge E2 is created with its head constituent instantiated with the feature specifications which resulted from the unification. This new edge E2 is annotated with the span of the inactive edge E1. Some time later the inactive edge E3 is matched against the "np" constituent of our active edge E2, resulting in the new active edge E4. The span of E4 is obtained by combining the starting position of E3 (i.e. 1) with the finishing postion of E2 (i.e. 3). The point is that edges are constructed from the head out, so that at any given time in the life cycle of an edge the spanning information on the edge records the span of contiguous substring which it covers.

Note that in the transition from rule R1 to edge E2 we have relabeled the constituent markers *x*, *c1*, and *h* with the symbols *s*, *np*, and *VP* respectively. This is done merely as a mnemonic device to reflect the fact that once the head of the edge is found, the subcategorization information on that head (i.e. the values of the "SUBCAT" feature of the verb *works*) is propagated to the other elements of the edge, thereby restricting the types of constituents with which they can be satisfied. Writing a constituent marker in upper case indicates that an inactive edge has been found to instantiate it, while a lower case (not yet found) constituent in bold face indicates that this is the next constituent which will try to be instantiated.

```
E1.  V<3,3>
R1.  x -> c1  h a*
E2.  s<3,3> ->  np VP a*

E3.  NP<1,2>
E2.  s<3,3> ->  np VP a*
E4.  s<1,3> -> NP VP a*
```

Figure 5: Combining edges and rules.

Using Semantics Restrictions

Parsing "head first" offers both practical and theoretical advantages. As mentioned above, the categories of the grammatical relations subcategorized for by a particular head are encoded as the SUBCAT value of the head. Now GR's are of two distinct types: those which are "saturated" (i.e. do not subcategorize for anything themselves), such as subject and objects, and those which subcategorize for a subject (i.e. controlled complements). One of the language-universal grammatical principles (the Control Agreement Principle) requires that the semantic controller of a controlled complement always be the next grammatical relation (in the order specified by the value of the SUBCAT feature of the head) after the controlled complement to combine with the head. But since the HPSG parser always finds the head of a clause first, the grammatical order of its complements, as well as their semantic roles, are always specified before the complements are found. As a consequence, semantic processing of constituents can be done on the fly as the constituents are found, rather than waiting until an edge has been completed. Thus semantic processing can be done extremely locally (constituent-to-constituent in the edge, rather than merely node-to-node in the parse tree as in Montague semantics), and therefore a parse path can be abandoned on semantic grounds (e.g. sortal inconsistency) in the middle of constructing an edge. In this way semantics, as well as syntax, can be used to control the parsing process.

Anaphora in HPSG

Another example of how parsing "head first" pays off is illustrated by the elegant technique this strategy makes possible for the binding of intrasentential anaphors. This method allows us to assimilate cases of bound anaphora to the same general binding method used in the HPSG system to handle other non-lexically-governed dependencies such as gaps, interrogative pronouns, and relative pronouns. Roughly, the unbound dependencies of each type on every constituent are encoded as values of an appropriate stack-valued feature

* An edge is, loosely speaking, an instantiation of a rule with some of the features on constituents made more specific.

("binding feature"). In particular, unbound anaphors are kept track of by two binding features, REFL (for reflexive pronouns) and BPRO for personal pronouns available to serve as bound anaphors. According to the Binding Inheritance Principle, all categories on binding-feature stacks which do not get bound under a particular node are inherited onto that node. Just how binding is effected depends on the type of dependency.

In the case of bound anaphora, this is accomplished by merging the relevant agreement information (stored in the REFL or BPRO stack of the constituent containing the anaphor) with one of the later GR's subcategorized for by the head which governs that constituent. This has the effect of forcing the node that ultimately unifies with that GR (if any) to be the sought-after antecedent. The difference between reflexives and personal pronouns is this. The binding feature REFL is not allowed to inherit onto nodes of certain types (those with CONTROL value INTRANS), thus forcing the reflexive pronoun to become locally bound. In the case of non-reflexive pronouns, the class of possible antecedents is determined by modifying the subcategorization information on the head governing the pronoun so that all the subcategorized-for GR's later in grammatical order than the pronoun are "contra-indexed" with the pronoun (and thereby prohibited from being its antecedent). Binding then takes place precisely as with reflexives, but somewhere higher in the tree.

We illustrate this distinction with two examples. In sentence S1 below *told* subcategorizes for three constituents: the subject NP *Pullum*, the direct object *Gazdar*, and the oblique object PP *about himself*.* Thus either *Pullum* or *Gazdar* are possible antecedents of *himself*, but not *Wasow*.

S1. Wasow was convinced that *Pullum* told *Gazdar* about *himself*.

S2. *Wasow* persuaded Pullum to shave *him*.

In sentence S2 *shave* subcategorizes for the direct object NP *him* and an NP subject eventually filled by the constituent *Pullum* via control. Since the subject position is contra-indexed with the pronoun, *Pullum* is blocked from serving as the antecedent. The pronoun is eventually bound by the NP *Wasow* higher up in the tree.

Heuristics to Optimize Search

The HPSG system, based as it is upon a carefully developed linguistic theory, has broad expressive power. In practice, however, much of this power is often not necessary. To exploit this fact the HPSG system uses heuristics to help reduce the search space implicitly defined by the grammar. These heuristics allow the parser to produce an optimally ordered agenda of edges to try based on words used in the sentence, and on constituents it has found so far.

One type of heuristic involves additional syntactic information which can be attached to rules to determine their likelihood. Such a heuristic is based on the currently intended use for the rule to which it is attached, and on the edges already available in the chart. An example of this type of heuristic is sketched below.

```
R1. x ->  c1 h a*

    Heuristic-1: Are the features of c1 +QUE?
```

Figure 6: A rule with an attached heuristic.

Heuristic-1 encodes the fact that rule R1, when used in its incarnation as the $S \rightarrow NP\ VP$ rule, is primarily intended to handle declarative sentences rather than questions. Thus if the answer to Heuristic-1 is "no" then this edge is given a higher ranking than if the answer is "yes". This heuristic, taken together with others, determines the rank of the edge instantiated from this rule, which in turn determines the order in which edges will be tried. The result in this case is that for a sentence such as S3 below, the system will prefer the reading for which an appropriate answer is "a character in a play by Shakespeare", over the reading which has as a felicitous answer "Richard Burton".

S3. Who is Hamlet?

It should be emphasized, however, that heuristics are not an essential part of the system, as are the feature passing principles, but rather are used only for reasons of efficiency. In theory all possible constituents permitted by the grammar will be found eventually with or without heuristics. The heuristics simply help a linguist tell the parser which readings are most likely, and which parsing strategies are usually most fruitful, thereby allowing the parser to construct the most likely reading first. We believe that this clearly differentiates HPSG from "ad hoc" systems which do not make sharp the distinction between theoretical principle and heuristic guideline, and that this distinction is an important one if the natural language understanding programs of today are to be of any use to the natural language programs and theories of the future.

ACKOWLEDGEMENTS

We would like to acknowledge the valuable assitance of Thomas Wasow and Ivan Sag in the writing of this paper. We would also like to thank Martin Kay and Stuart Shieber for their helpful commentary on an earlier draft.

* The preposition is treated essentially as a case marking.

REFERENCES

[1] **Bresnan, J.** (ed). (1982)
The Mental Representation of Grammatical Relations, The MIT Press, Cambridge, Mass.

[2] **Creary, L. and C. Pollard** (1985)
"A Computational Semantics for Natural Language", *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics.*

[3] **Dowty, D.R.** (1982)
"Grammatical Relations and Montague Grammar", In P. Jacobson and G.K. Pullum (eds.), *The Nature of Syntactic Representation* D. Reidel Publishing Co., Dordrecht, Holland.

[4] **Earley, J.** (1970)
"An efficient context-free parsing algorithm", CACM 6:8, 1970.

[5] **Flickinger, D., C. Pollard, T. Wasow** (1985)
"Structure-Sharing in Lexical Representation", *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics.*

[6] **Gawron, J. et al.** (1982)
"Processing English with a Generalized Phrase Structure Grammar", *ACL Proceedings 20.*

[7] **Gazdar, G. et al.** (in press)
Generalized Phrase Structure Grammar, Blackwell and Harvard University Press.

[8] **Kaplan, R.** (1973)
"A General Syntactic Processor", In Rustin (ed.) *Natural Language Processing.* Algorithmics Press, N.Y.

[9] **Kay, M.** (1973)
"The MIND System", In Rustin (ed.) *Natural Language Processing.* Algorithmics Press, N.Y.

[10] **Kay, M.** (forthcoming)
"Parsing in Functional Unification Grammar".

[11] **Pollard, C.** (1984)
Generalized Context-Free Grammars, Head Grammars, and Natural Language, Ph.D. Dissertation, Stanford.

[12] **Pollard, C.** (forthcoming)
"A Semantic Approach to Binding in a Monostratal Theory", To appear in *Linguistics and Philosophy.*

[13] **Winograd, T.** (1980)
Language as a Cognitive Process.
Addison-Wesley, Reading, Mass.