Quick Notes on HPSG

Nick Fleisher
fleisher@berkeley.edu

January 31, 2007

1 Fundamentals

For an overview of the basic theoretical commitments and design principles of HPSG,[1] see the concise and useful summary posted at http://hpsg.stanford.edu/ideas.html

HPSG is a UNIFICATION-based theory of grammar, which means that structures must satisfy specified CONSTRAINTS on their forms all at once, rather than via a syntactic derivation. In place of the generative-Minimalist notion of a syntactic derivation proceeding in steps, in HPSG all grammatical constraints are satisfied simultaneously at one level of grammatical structure. HPSG is thus a MONOSTRATAL theory of grammar.

The basic unit of linguistic representation in HPSG is the SIGN. Signs contain information about the phonology, syntax, and semantics of a linguistic unit. Each type of information within a sign is called an ATTRIBUTE (also known as a FEATURE), and each attribute must have a VALUE (see "Attribute-Value Matrices" below). Constraints in HPSG typically take the form of ensuring that a value is shared by two or more attributes within a sign, or that values for particular attributes are shared by multiple signs. This last type of constraint is central to the notion of headedness within HPSG, as a phrasal sign (i.e., XP) must share certain information—about its syntactic category, its agreement properties, etc.—with its head (X) (usually called the HEAD-DAUGHTER).

2 Attribute-Value Matrices

HPSG notation can be a bit opaque at first. Attribute-value matrices (AVMs) are used to represent signs. By convention, attributes are set in small caps, values in italics. Square brackets set off attributes within a given hierarchy of attribute types. Angle brackets enclose ordered lists (i.e., tuples) of values. Examples (20) and (21) from Bouma et al. (2001) are given here:

(20) SLASH Amalgamation

$$word \Rightarrow \left[\begin{array}{l} \textsc{loc}\left[\textsc{cat}\left[\begin{array}{ll}\textsc{deps} & \left\langle\left[\textsc{slash}\ \boxed{1}\right], \ldots, \left[\textsc{slash}\ \boxed{n}\right]\right\rangle \\ \textsc{bind} & \boxed{0}\end{array}\right]\right] \\ \textsc{slash}\left(\boxed{1} \cup \ldots \cup \boxed{n}\right) - \boxed{0}\end{array}\right]$$

[1] Head-driven Phrase Structure Grammar.

(21)
$$\begin{bmatrix} \text{I-FORM} & \textit{know} \\ \text{SUBJ} & \langle \boxed{3} \rangle \\ \text{COMPS} & \langle \boxed{4} \rangle \\ \text{DEPS} & \left\langle \boxed{3}\begin{bmatrix} \text{LOC} & \text{NP} \\ \text{SLASH} & \boxed{1} \end{bmatrix}, \boxed{4}\begin{bmatrix} \text{LOC} & \text{S}[\textit{fin}] \\ \text{SLASH} & \boxed{2} \end{bmatrix} \right\rangle \\ \text{BIND} & \{\} \\ \text{SLASH} & \boxed{1} \cup \boxed{2} \end{bmatrix}$$

In (20), we see a constraint regarding SLASH values; this is the SLASH amalgamation constraint. Here is what it means: Each item in the DEPS list has a SLASH attribute with a particular value (this is the only attribute for these items shown here, but this is only for convenience; in reality, these items are whole signs). The SLASH attribute for the entire *word*, shown at the bottom of the AVM, is constrained to have as its value the union of the SLASH values of all the items in the DEPS list (i.e., 1 through n), minus any SLASH value that is bound by the value of the BIND attribute (in this case, $\boxed{0}$).

We can see this constraint at work again in (21), where the value of SLASH at the bottom of the AVM (i.e., for the sign whose I-FORM is *know*) is the union of the SLASH values of the items on the DEPS list. We can also see the operation of a constraint relating the values of DEPS to the values of SUBJ and COMPS, as indicated by the indices $\boxed{3}$ and $\boxed{4}$. Why don't SUBJ and COMPS show the same mini-AVMs as we see in the DEPS list? The answer is that we are free to show these AVMs as the values of SUBJ and COMPS, but to do so would be redundant, since they are already shown once in DEPS and indexed there. Indexing ensures that the values of SUBJ and COMPS are exactly the same as the corresponding values in the DEPS list.

3 The Boxed-Number Thing

AVMs are full of boxed numbers. These are simply indices; they can represent either values for attributes or entire signs. They are used in HPSG as a convenient shorthand for the kind of feature matching / sharing that characterizes most HPSG constraints. It is important to keep in mind that boxed-number indices may represent null values. This is particularly common with the SLASH attribute involved in the analysis of unbounded dependencies. Thus, the SLASH attribute in (21) whose value is $\boxed{1} \cup \boxed{2}$ may very well represent the empty set, as it is perfectly possible for the SLASH values on both items in the DEPS list—the values whose union is being taken in the bottom SLASH value—to be empty sets.[2] Note as well in this connection that the index $\boxed{0}$ need not represent the empty set; the numbers within indices are completely arbitrary.

4 SLASH Termination

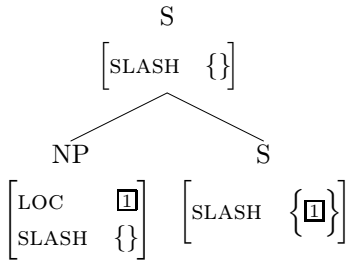
Of particular importance for understanding unbounded dependencies within HPSG is the notion of SLASH termination. This is the process that regulates the grammatical connection between a gap and its filler (described briefly on p. 19 of Bouma et al.). Essentially, SLASH values represent gaps; when the LOCAL value of one sign matches the SLASH value of its sister, this causes the SLASH value of their

[2] SLASH, as you may recall from the article, is a SET-VALUED attribute, meaning that its value is a set, and not, e.g., a single value or a tuple.

mother to be empty.[3] LOCAL attributes are ones that encode basic intrinsic properties of a sign, such as its syntactic category, person, number, and many others. SLASH is a NON-LOCAL attribute, and it must be passed up from daughters to mothers through a tree, until it is terminated by unification with a sister whose LOCAL value matches it. The is the essence of how unbounded dependencies are treated in HPSG. Note that, since the LOC and SLASH attributes related by SLASH termination are constrained to have the same values, the theory ensures that fillers have the grammatical properties expected of elements that (would) occur in their corresponding gap sites.

To see SLASH termination in action, consider the top of the tree in Bouma et al.'s example (24):

(24)



The NP and S that are the immediate daughters of the root S have matching LOC and SLASH values, respectively (i.e., the value 1). This allows the LOC value of NP to unify with the SLASH value of its sister S, yielding an empty SLASH value for the mother S. As Bouma et al. put it, "The missing element is 'found' at this point, and thus the SLASH of the resulting phrase is empty" (p. 19). Unbounded dependencies are thus treated in HPSG as a combination of SLASH-value passing from daughters to mothers, and SLASH termination (i.e., unification) with a sister whose LOC value matches the relevant SLASH value.

5 Summary

We haven't attempted any meaningful discussion here of how larger phrase structures are built, how signs are internally structured, or how the complex hierarchy of types helps determine the feature structure of particular signs. I simply hope to have given you a very basic idea of the general model of grammar in HPSG, along with some pointers on how to read the notation. From this, you should be able to delve deeper on your own.

[3] Provided that the 'filler' sign's own SLASH value is empty.