Lectures on the Foundations of HPSG

Carl Pollard
Ohio State University

1 General character of HPSG

1.1 Similarities to Chomsky's EST/GB Framework

1.1.1 Same principal goal: To characterize human linguistic competence

i.e., to construct a scientific theory of the system of knowledge that is embodied in the human mind/brain which makes language possible. This is distinct from constructing a psycholinguistic theory (a theory of how the mind/brain uses that knowledge to produce and interpret utterances). Knowledge of language is like a database or a system of knowledge representation; it is not a set of algorithms or procedures for processing language (though such processes must have access to the system of linguistic knowledge: see below on "psycholinguistic responsibility."). Part of the goal of characterizing competence is determining what all languages have in common (so-called "Universal Grammar").[1]

1.1.2 Same empirical base: acceptability judgements of speakers

However, care is taken to consider judgements only relative to specified classes of contexts. It is important to control for differing abilities of native speakers to imagine contexts in which a given string becomes acceptable.

1.1.3 Multiple representations

more or less analogous to "levels" of representation in EST/GB. But these are simultaneous (or coexisting, or parallel) structures that are mutually constrained by the grammar; all are parts of a single larger structures and none is sequentially derived from another.

1.1.4 Grammaticality is determined by the interaction between the lexicon and general well-formedness principles

(roughly as in GB) rather than by large numbers of construction-specific rules (as in Chomsky's "standard theory", classical Montague grammar, or GPSG).

1.1.5 Many EST/GB concepts have HPSG analogs.

Examples inlcude θ-roles, indices, agreement features, and traces. But in HPSG these are all carefully formalized so that empirically vulnerable predictions can be made.

[1] Transformational grammar has always made a point of being concerned with limiting the range of possible variation across languages and explaining how language can be acquired. HPSG researchers consider these to be interesting long-term goals, but don't think enough is known yet to warrant positing any empirical hypotheses.

1.1.6 Likewise many EST/GB principles have rough HPSG analogs.

Examples include binding principles A, B, and C, and constraints on "extraction" (such as subjacency and ECP).

1.2 Sociological differences

1.2.1 No "fearless leader"

There is no Chomsky-like figure who is always assumed to be basically on the right track no matter what s/he proposes. HPSG research is normal science: the testing of hypotheses that appear plausible given accepted assumptions. The goal is not to fill in the details of a vague theory which is assumed to be basically right, but to successively replace empirical hypotheses with ones that make better predictions.

1.2.2 Receptivity to adjacent technology

HPSG was developed in an environment where some familiarity with such technical bodies of knowledge as logic, set theory, algebra, graph theory and/or theoretical computer science could be assumed. Practitioners are willing to acquire a wide range of technical tools, and apply them in the interest of making their analyses and theories clearer and more precise. Consequently, they can have more confidence in their proposals.

1.3 Methodological differences

1.3.1 Freedom from typological bias

It is not assumed that all languages are basically like English. Languages are not assumed to vary without limit, but the starting assumption is that we do not know what the range of variation is, and we can take very little for granted when we begin to try to uncover the structure of a language.

1.3.2 Nonprimacy of syntax

HPSG is not syntactocentric. There is no assumption that syntax is somehow primary, and that morphology is done "in the syntax" (cf. affix hopping, head movement). Phonology and semantics are not "interpretations" of syntactic structures as in transformational models from *Syntactic Structures* to the Minimalist Program. Instead, it is assumed that we need to understand several different systems of linguistic knowledge, including syntax, semantics, morphology, phonology/prosody, and pragmatics/discourse. In HPSG theories, most grammatical principles don't involve just one of these, but instead constrain the relationship between two or more.

1.3.3 Empirical adequacy

HPSG is done bottom-up or inductively, generalizing from specifics instead of starting with grand generalities and looking for particulars that confirm them. Thus it employs the fragment methodology: make a precise, falsifiable hypothesis that accounts for a wide range of facts (i.e. get the details right for a subpart of the language), then revise the hypothsis to expand coverage. Precise, empirically vulnerable generalizations with broad, determinable consequences (which might not follow from very deep principles) are valued over deep principles that are so vague that their empirical consequences cannot be deduced.

1.3.4 Psycholinguistic responsibility

Grammars, as models of competence, should nonetheless be capable in principle of being exploited by plausible psycholinguistic models. Thus, they should not be based IRREDUCIBLY on computations that a language user would be incapable of carrying out. And thus there is a somewhat different emphasis in the subject matter: instead of a (largely rhetorical) concern with how linguistic knowledge is acquired, rather one is concerned with how it is organized so as to make language use possible. The language user is viewed more as an information-processing agent than as language-acquisition device. Ultimately, grammatical knowledge must be organized in such a way that it can be employed efficiently in a wide variety of tasks, including production, understanding, translation, language games, making acceptability judgements, and more. But the linguistic framework itself should be TASK-NEUTRAL.

1.3.5 Generativity

HPSG research takes it for granted (as Chomsky originally did) that the goal of a grammatical theory of language X is minimally to tell what the well-formed structures of language X are ("descriptive adequacy"). Thus, formal precision is necessary (the current Chomskyan view is that it is "premature" and therefore undesirable). In an adequate formalization, according to Pullum (1989):

1. It must be made clear what mathematical structures are used to model different kinds of linguistic entities. (In HPSG, the mathematical structures are graphs of a certain kind called feature structures.)

2. It must be determinate what the actual assertions of the theory are. They don't have to be framed in formal logic or Prolog or Lisp or C++ though sometimes it is helpful to do this. Careful English (or Korean, etc.) will do, as long as it is clear, specific, and unambiguous.

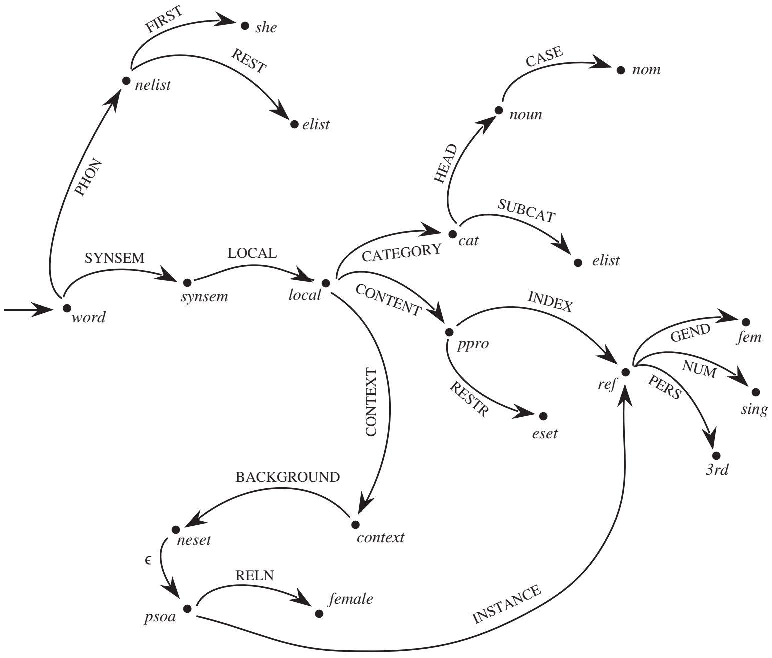
3. Given a grammar G and a mathematical object O used as a model of a candidate linguistic entity (a representation), there has to be a way to tell whether or not O satisfies the constraints imposed by G.

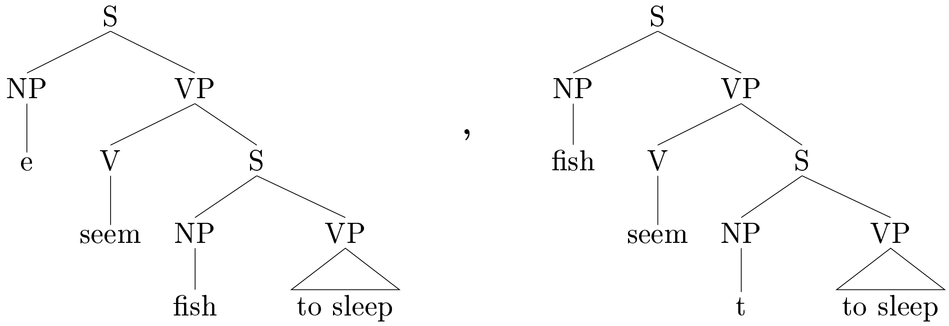
1.4 Architectural differences

There are large-scale differences in how theories are formulated or organized.

1.4.1 Structures employed as mathematical idealizations of linguistic entities

HPSG employs different mathematical structures from EST/GB. The mathematical structures of transformational theories are sequences of phrase-markers (trees, i.e., rooted, directed, graphs satisfying the single mother condition, with nodes labelled by category symbols).





In HPSG, the mathematical structures are feature structures: rooted, connected, directed graphs, with each arc labelled by a feature name, and each node labelled by a species name (the name of a kind of linguistic object).

- A **rooted** graph is one that has a distinguished node called the **root**.
- In a **connected** graph, every node is reachable from the root by a finite sequence of edges.
- In a **directed** graph, the edges are directed (i.e. they are ordered pairs of nodes, not just doubleton sets of nodes).

1.4.2 Grammars are formalized

Fully formalized, an HPSG grammar is formulated as a set of well-formedness constraints on feature structures, of which each constraint is a (nonlogical) axiom in a certain kind of formal language called a feature logic. (In these notes, we use a slight notational variant of King's (1989) SRL). The familiar attribute-value matrices (AVMs) are an INFORMAL substitute for feature logic constraints. Feature structures and feature logic satisfy the criteria of formal precision stated above in subsection 1.3.5. The constraints tell which feature structures are well-formed representations of linguistic entities. Technically, a grammar is a logical theory, and a well-formed structure is a model of the theory.

The feature logic is itself unconstrained in terms of what constraints it can express. It is like predicate calculus or LISP in this respect: the constraints don't come from limits on what theories the logic can express; they ARE a theory that the logic expresses. Thus HPSG employs an EXPRESSIVE formalism, not a constrained formalism.

1.4.3 Nonderivationality

HPSG is nonderivational. It employs PARALLEL REPRESENTATIONS which are MUTUALLY CONSTRAINED by the grammar. This is also true of other nontransformational frameworks, such as LFG, APG, autolexical syntax, construction grammar, and Jackendoff's (1996) programmatic Representational Modularity framework.

HPSG employs no transformations or other destructive operations to sequentially derive one structure (or representation) from another. Instead, the different representations (or levels, in Ladusaw's (1988) terminology) are just subparts (features, actually) of a single larger structure, and they are related not by destructive operations, but rather by declarative (asserted) constraints of the grammar. [2] (Thus, the substructure reached from the root by following the PHON path is a rough correspondent of the GB level PF; and the one reached by the path SYNSEM|LOC|CAT in a verb corresponds roughly to the DS of the sentence headed by that verb; the substructure reached by the path SYNSEM|LOC| CONT corresponds roughly to the GB level LF. Understood this way, HPSG has considerably more levels than GB.)

1.4.4 Structural uniformity

HPSG is "fractal" (structurally uniform as the parts get smaller). Every sign down to the word level (not just the root clause) has features corresponding (inter alia) to phonetic, syntactic, and semantic aspects of linguistic structures

1.4.5 No lexical insertion

Words are just the same kind of thing (namely, signs) as phrases; they are just small-scale constrained parallel structures. Thus, there is no distinction between terminal nodes and preterminals, and no issue (as in recent Minimalist Program work) of whether lexical insertion is "early" or "late". This is a consequence of the theory being nonderivational and fractal.

1.4.6 Locality

HPSG employs only local constraints. There are no global constraints constraining structures relative to other structures. Well-formedness is determined completely with reference to a given structure, and not via comparison with any other "competing" structure (as it might be in MP and OT).

[2] IRREDUCIBLE use of transformations is hard to reconcile with psycholinguistic responsibility: to determine PF from LF, an SS must first be determined, from which one could work back to a DS source, and ahead to derive a PF. The first two steps require "inverting transformations," which is a computational nightmare. Similarly for determining LF from PF. Although some GBists (e.g. Koster (1987), Brody (1995)) have argued that GB theory could just as well be reformulated without transformations, Chomsky has never been sympathetic to this position, and with the Minimalist Program, it is not really an option.

1.5 Technical Differences

There are also smaller-scale differences in mechanisms employed by the framework. Some representative examples:

- Tree-configurational notions like c-command and government have no role in HPSG, which employs instead the traditional relation of **obliqueness of grammatical relations**, where obliqueness increases from *Subject* to *Primary Object* to *Secondary Object* to *Oblique PPs* and VP, AP, and S *Complements*. Thus, e.g. Principle A of the binding theory requires not that an r-pronoun (anaphor) be coindexed with a c-commanding NP in some local domain, but rather that it be coindexed with a less oblique argument of the same head (if there is one).

- **Nonconfigurational definitions of grammatical relations**. Grammatical relations are defined NONconfigurationally, in terms of their relative order on the lists that are the values of the VALENCE features. Thus, the subject is the unique item on the SUBJ list; the primary object is the first item on the COMPS list, etc., rather than being sister-of-VP-and-daughter-of-S, or daughter-of-VP-and-closest-sister-of-V, as in transformational theories from *Aspects* through GB.

- **Selection of subjects and specifiers**. Subjects and specifiers (and their detailed properties, such as number and definiteness) can be selected ("subcategorized for") by lexical heads in HPSG, just as complements are. Thus subjectless sentences are just as possible as intransitive verbs, since a verb can select for "no subject".

- **Semantic roles** ("θ-roles") are assigned to subjects directly, in the same way as to complements, not indirectly.

- Likewise, **either subject or object can fail to be assigned a semantic role**. Among other things, this means that "raising-to-object" analyses are not ruled out. In GB, by contrast, nonthematic objects are impossible.

- There is **no requirement that every VP have a subject**; thus, verbs that appear to take infinitive VP complements can be treated as actually doing so, with no need for a phonologically null subject (PRO or NP-trace). Control is treated as coindexing of the controller phrase with the value of the SUBJ valence feature of the VP complement.

The following differences are directly related to the nonderivationality of HPSG.

- **No NP-movement** (and, more generally, no movement at all.)

  Passive: Passive verbs are in the lexicon (perhaps mostly derived by lexical rule). They project a phrase according to the same constraints as active verbs do.

  Raising: STRUCTURE-SHARING between the NP subject or object and the member of the SUBJ valence feature list of the VP complement takes the place of NP-movement. Thus, the analysis of raising structures differs from the analysis of equi-structures only in that 1) raising predicates systematically have one semantic role fewer than equi predicates with the same number of syntactic arguments, and 2) raising verb select (as subject or object) for the same structure that the VP complement selects for its subject, while in the case of an equi verb, the subject or object is only coindexed with the complement subject. Moreover, raising to subject and raising to object are treated on a par.

- **No WH-movement**: The value of the SLASH feature throughout the path from the verb whose argument is "missing" to the filler WH-element is structure-shared with (1) the LOCAL value of the filler element, and (2) the LOCAL value of the missing argument.

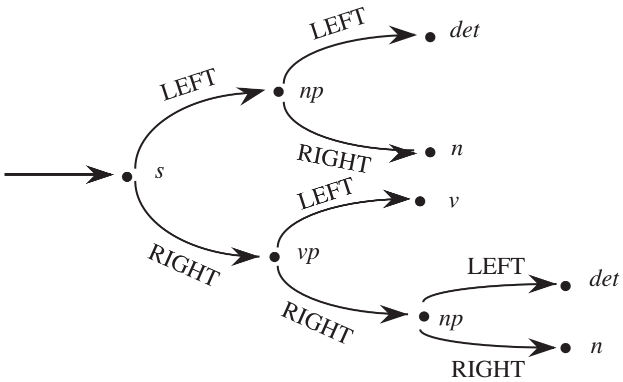
- **No head movement**:

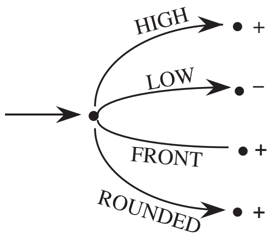
  Verb inflection: finite verbs are specified in the lexicon to select subjects bearing specific agreement features. Thus there is no movement between V and Infl (in fact there is no need for a distinct Infl node).

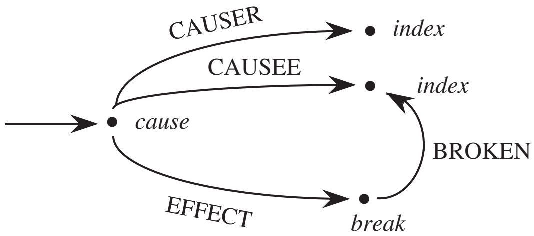
  Inversion: Inverted structures are "flat" structures, not derived from uninverted structures. Thus there is no movement from Infl to Comp.

- More generally, there are **no null functional heads** (T, AgrO, AgrS, etc.). Instead the corresponding work in HPSG is done by features.

- Likewise, there are **no null complementizers**.







2 Foundations of the HPSG Formalism

2.1 Feature structures

Feature structures are mathematical objects—graphs of a certain kind—that are used as theoretical models of structured entities of all sorts in the real world, not just linguistic entities. Thus, the diagrams above represent a high front rounded object, a constituent structure diagram, and the relation of breaking. (Note: these are for illustrative purposes only. They are not being posited as serious linguistic analyses.)

3

2.2 Formal Machinery

Assume that there are given, disjoint, finite sets $\mathcal{F}$ of *feature names*, and $\mathcal{S}$ of *species names*.

Definition: A **feature graph** is an ordered triple $\mathcal{G} = <\mathcal{U}$, S, F $>$, where:

a. $\mathcal{U}$ is a set (called the *nodes* of $\mathcal{G}$);

b. F is a function which associates with each feature name f a partial function F(f) (also written $f_{\mathcal{G}}$) from $\mathcal{U}$ to $\mathcal{U}$; and

c. S is a function which associates with each species name s a subset of $\mathcal{U}$; S(s) is also written $s_{\mathcal{G}}$).

Mathematically, a feature graph is just a sorted unary partial algebra, where S interprets the sorts (species names) and F interprets the operations (feature names).

There is also a more primitive notion of (directed) graph, where all we have is nodes and (directed) edges (with no species or features). This is defined as follows:

Definition: A **(directed) graph** is an ordered pair $\mathcal{G} = < \mathcal{U}, \rightarrow >$, where:

a. $\mathcal{U}$ is a set (called the *nodes* of $\mathcal{G}$);
b. $\rightarrow$ is a binary relation on $\mathcal{U}$ (i.e., a subset of $\mathcal{U} \times \mathcal{U}$) called the *edges* of $\mathcal{G}$. (We write $p \rightarrow q$ to mean $< p,\ q > \in \rightarrow$.)

Often in working with graphs we single out a node which serves as a sort of "home base". Technically:

Definition: A **pointed graph** is an ordered triple $\mathcal{G} = < q_0, \mathcal{U}, \rightarrow >$, where $< \mathcal{U}, \rightarrow >$ is a graph (called the *underlying* graph of $\mathcal{G}$) and $q_0 \in \mathcal{U}$; q_0 is called the *point* of $\mathcal{G}$.

Definition: Given a graph $< \mathcal{U}, \rightarrow >$ and two nodes $p,\ q \in \mathcal{U}$, we say q is **accessible** from p just in case $p \rightarrow^* q$. (For any binary relation R, R* is the reflexive transitive closure of R.)

Informally, q is accessible from p just in case there is a path (finite sequence of edges) from p to q.)

Definition: A pointed graph is called **accessible** (or an **apg**) iff every node is accessible from the point. In this case, the point is called the **root** of the apg.

Thus, if $\mathcal{G} =< q_0, \mathcal{U}, \rightarrow >$ is an apg, then q_0 is the root and every node in $\mathcal{U}$ is accessible from q_0. Trees are a familiar kind of apg. (An apg is a tree just in case each node is accessible from the root via a unique path.)

Observation: Let $\mathcal{G} =< \mathcal{U}$, S, F $>$ be a feature graph, and let $\rightarrow_{\mathrm{F}} \subseteq \mathcal{U} \times \mathcal{U}$ be defined as $\bigcup\{f_{\mathcal{G}} : f \in \mathcal{F}\}$. Then $< \mathcal{U}, \rightarrow_{\mathrm{F}} >$ is a graph.

That is, given a feature graph, we can form the graph whose set of edges is the union of the interpretations of the feature names. Intuitively, this amounts to throwing away all the labels on the nodes and edges.

Definition: A **feature structure** is a quadruple $< q_0, \mathcal{U}, \mathcal{S}, \mathcal{F} >$, where $< \mathcal{U}, \mathcal{S}, \mathcal{F} >$ is a feature graph, and $< q_0, \mathcal{U}, \rightarrow_{\mathrm{F}} >$ is an apg. We call q_0 the **root** of the feature structure.

Feature structures are what we will use as our mathematical idealizations of linguistic entities (i.e. analyses of expressions).

2.3 Graph Notation

Nodes are represented by dots, and edges (or arcs) by arrows. Thus, in the graph $\bullet \longrightarrow \bullet$, if the left dot is p and the right dot is q, then $p \rightarrow q$. For a feature graph, an arrow labelled with a feature name f means that $f_{\mathcal{G}}(p) = q$, where p is represented by the dot from which the arrow originates and q is represented by the dot the arrow points to. The root of a feature structure is indicated by a short boldface arrow to a node: $\boldsymbol{\rightarrow} \bullet$.

2.4 Feature structures generated by nodes in feature graphs

Definition: Let $\mathcal{G} = < \mathcal{U}, \mathrm{S}, \mathrm{F} >$ be a feature graph and $q \in \mathcal{U}$ a node. The feature structure **generated by** q **in** $\mathcal{G}, \mathcal{G}_q$, is $< q, \mathcal{U}_q, \mathrm{S}_q, \mathrm{F}_q >$, where:

a. $\mathcal{U}_q = \{p \in \mathcal{U} \mid q \rightarrow^*_{\mathcal{F}} p\}$;
b. for each s in $\mathcal{S}, \mathrm{S}_q(s) = s_{\mathcal{G}} \cap \mathcal{U}_q$;
c. for each f in $\mathcal{F}, \mathrm{F}_q(f) = f_{\mathcal{G}} \cap (\mathcal{U}_q \times \mathcal{U}_q)$.

Thus the feature structure generated by a node q in a feature graph $\mathcal{G}$ contains only the nodes of $\mathcal{G}$ accessible from q; and the interpetations of species names and feature names are obtained by restricting their interpretations in $\mathcal{G}$ in the obvious way.

2.5 Feature structure homomorphisms and isomorphisms

Let $\mathcal{G} = < q_0, \mathcal{U}, \mathrm{S}, \mathrm{F} >$ and $\mathcal{G}' = < q'_0, \mathcal{U}', \mathrm{S}', \mathrm{F}' >$.

Definition: A **homomorphism** from $\mathcal{G}$ to $\mathcal{G}'$ is a function $\eta : \mathcal{U} \rightarrow \mathcal{U}'$ such that

a. $\eta(q_0) = q'_0$;
b. for all $q \in \mathcal{U}$ and $f \in \mathcal{F}$, if $f_{\mathcal{G}}$ is defined at q, then $f_{\mathcal{G}'}$ is defined at $\eta(q)$, and $f_{\mathcal{G}'}(\eta(q)) = \eta(f_{\mathcal{G}}(q))$;
c. for all $q \in \mathcal{U}$ and $s \in \mathcal{S}$, if $q \in s_{\mathcal{G}}$, then $\eta(q) \in s_{\mathcal{G}'}$.

This is just like a homomorphism in algebra (thinking of the interpretations of the feature names as the algebra operations), with the additional provisos that the species and the root must be preserved.

Definition: A homomorphism is called an **isomorphism** if it has an inverse which is also a homomorphism. It follows that isomorphisms are both one-to-one and onto.

Thus isomorphic feature structures "look the same"; they just have different nodes. It is easy to see that, for any feature structure $\mathcal{G} = < q_0, \mathcal{U}, \mathrm{S}, \mathrm{F} >$, the identity function on $\mathcal{U}, id_{\mathcal{U}}$, is an isomorphism from $\mathcal{G}$ to $\mathcal{G}$. Also, if η is a homomorphism from $\mathcal{G}$ to $\mathcal{G}'$ and η' is a homomorphism from $\mathcal{G}'$ to $\mathcal{G}''$, then $\eta' \circ \eta$ is a homomorphism from $\mathcal{G}$ to $\mathcal{G}''$.

In HPSG, as explained below, a grammar is a kind of logical theory whose intended interpretations are feature structures. Thus the grammar picks out a certain class of feature structures, namely the ones that are models of the grammar. Now clearly, for any feature structure, there are infinitely many other feature structures (in fact, a proper class of them) isomorphic to it. However, as linguists, when we are given a grammar, we don't really care about the whole class of models. All we really care about is which feature structures are models *up to isomorphism*; e.g. the fact that there is an infinity of different feature structure models of the sentence *poor John ran away* is not of interest to us, since all these models are isomorphic. What we would really like is a way to pick out from each isomorphism class of models a unique canonical representative. Then we could consider the grammar to **generate** precisely the set of feature structures which are the canonical representatives of the various isomorphism classes of models. This idea connects the model-theroetic interpetation of grammars with the standard linguistic notion of (strong) generative capacity.

In fact, there is a rather simple mathematical technique for constructing such representative feature structures; roughly speaking, the trick is to choose feature structures whose nodes are equivalence classes of paths (strings of feature names). The precise construction, which is a slight variant of Moshier's (1988) notion of an *abstract* feature structure, is described in Pollard 1998.

2.6 Paths

A **path** is a member of $\mathcal{F}^*$, i.e., it is a finite sequence (string) of feature names. As usual, the null path (path of length zero) is denoted by ϵ. Given a feature graph $\mathcal{G} =< \mathcal{U}, \mathrm{S}, \mathrm{F} >$, the function $\mathrm{F} : \mathcal{F} \rightarrow [\mathcal{U} \rightharpoonup \mathcal{U}]$ can be extended to a function $\tilde{\mathrm{F}} : \mathcal{F}^* \rightarrow [\mathcal{U} \rightharpoonup \mathcal{U}]$ as follows: $\tilde{\mathrm{F}}$ is defined by recursion on path lengths n such that:

a. if a path π is of length 0 (i.e., $\pi = \epsilon$) then $\tilde{\mathrm{F}} = id_{\mathcal{U}}$;

b. if π is of length n > 0 and $\pi = f\pi'$ where $f \in \mathcal{F}$ and π' is a path of length n-1, then

$$\tilde{\mathrm{F}}(\pi) = (\tilde{\mathrm{F}}(\pi')) \circ (F(f)).$$

(That is, for a path of length n > 0, the function it denotes is the composition of two functions: the one denoted by its final subpath of length n-1, and the one denoted by its initial feature.)

$\tilde{\mathrm{F}}(\pi)$ is also written $\pi_{\mathcal{G}}$. Thus

$$\epsilon_{\mathcal{G}} = id_{\mathcal{U}}$$

and if $\pi = f\pi'$, then

$$\pi_{\mathcal{G}} = \pi'_{\mathcal{G}} \circ f_{\mathcal{G}}$$

Also, $(f_1 \cdots f_n)_{\mathcal{G}} = (f_n)_{\mathcal{G}} \circ \cdots \circ (f_1)_{\mathcal{G}}$.

2.7 King formulas

Given disjoint finite sets $\mathcal{F}$ (feature names) and $\mathcal{S}$ (species names), the set $\mathcal{K}$ of **King formulas** (cf. King (1989) is defined recursively as follows:

a. for each s in $\mathcal{S}$, $s \in \mathcal{K}$;

b. $\top \in \mathcal{K}$;

c. for each path π and each formula $\phi \in \mathcal{K}, (\pi : \phi) \in \mathcal{K}$

d. for π_1 and π_2 paths $(\pi_1 \doteq \pi_2) \in \mathcal{K}$;

e. for π_1 and π_2 paths $(\pi_1 \not\doteq \pi_2) \in \mathcal{K}$;

f. for ϕ and $\psi \in \mathcal{K}, (\phi \wedge \psi) \in \mathcal{K}$;

g. for ϕ and $\psi \in \mathcal{K}, (\phi \vee \psi) \in \mathcal{K}$;

h. for ϕ and $\psi \in \mathcal{K}, (\phi \rightarrow \psi) \in \mathcal{K}$;

i. for ϕ and $\psi \in \mathcal{K}, (\neg\phi) \in \mathcal{K}$;

j. nothing else is in $\mathcal{K}$.

As described below, King formulas are used in HPSG both as descriptions of feature structures and as constraints in grammars. To explain the difference, we begin by defining a **satisfaction** relation $\models$ between feature structures and King formulas, analogous to the relation between first-order models and first-order formulas.

2.8 Satisfaction

The satisfaction relation $\models$ between a feature structure $\mathcal{G} = < q_0, \mathcal{U}, \mathrm{S}, \mathrm{F} >$ and a King formula is defined by structural recursion on formulas as follows:

a. $\mathcal{G} \models s$ iff $q_0 \in s_{\mathcal{G}}$;

b. $\mathcal{G} \models \top$ never;

c. $\mathcal{G} \models (\pi : \phi)$ iff $\pi_{\mathcal{G}}$ is defined at q_0 and $\mathcal{G}_{\pi_{\mathcal{G}}(q_0)} \models \phi$;

d. $\mathcal{G} \models (\pi_1 \doteq \pi_2)$ iff $(\pi_1)_{\mathcal{G}}$ and $(\pi_2)_{\mathcal{G}}$ are both defined at q_0 and have the same values;

e. $\mathcal{G} \models (\pi_1 \not\doteq \pi_2)$ iff $(\pi_1)_{\mathcal{G}}$ and $(\pi_2)_{\mathcal{G}}$ are both defined at q_0 and have different values;

f. $\mathcal{G} \models (\phi \wedge \psi)$ iff $\mathcal{G} \models \phi$ and $\mathcal{G} \models \psi$;

g. $\mathcal{G} \models (\phi \vee \psi)$ iff $\mathcal{G} \models \phi$ or $\mathcal{G} \models \psi$;

h. $\mathcal{G} \models (\phi \rightarrow \psi)$ iff $\mathcal{G} \not\models \phi$ or $\mathcal{G} \models \psi$;

i. $\mathcal{G} \models (\neg\phi)$ iff $\mathcal{G} \not\models \phi$.

Note: if $\mathcal{G}$ and $\mathcal{G}'$ are isomorphic feature structures, ϕ is a King formula, and $\mathcal{G} \models \phi$, then also $\mathcal{G}' \models \phi$.

If $\mathcal{G} \models \phi$, then we say $\mathcal{G}$ **satisfies** ϕ or ϕ **describes** $\mathcal{G}$. A formula is called **satisfiable** if some feature structure satisfies it (or, equivalently, if it describes some feature structure). For a set of formulas Φ, we say $\mathcal{G} \models \Phi$ iff $\mathcal{G} \models \phi$ for every $\phi \in \Phi$.

2.9 Entailment and (semantic) equivalence

Let Φ be a set of formulas and ϕ a formula. Then we say Φ **entails** ϕ, written $\Phi \models \phi$, provided for all feature structures $\mathcal{G}$, if $\mathcal{G} \models \Phi$ then $\mathcal{G} \models \phi$. If Φ contains only one formula ϕ, we usually write $\phi \models \psi$ for $\Phi \models \psi$. If ϕ and ψ are two formulas such that $\phi \models \psi$ and $\psi \models \phi$, then we say ϕ and ψ are (semantically) equivalent, written $\phi \approx \psi$. Equivalent formulas describe the same feature structures.

2.10 Constraints and Grammars

Let ϕ be a formula and $\mathcal{G}$ a feature structure (or more generally, a feature graph). Then we say $\mathcal{G}$ **models** ϕ (or, is a **model** of ϕ) provided, for every node q of $\mathcal{G}, \mathcal{G}_q \models \phi$. For Φ a set of formulas (e.g., a theory), we say $\mathcal{G}$ **models** Φ if $\mathcal{G}$ models ϕ for every $\phi \in \Phi$. Note: if $\mathcal{G}$ models ϕ, then $\mathcal{G} \models \phi$, but the converse is in general false.

It is important to understand the relation between a *description* and a *constraint*. A formula is called a **description** when we are concerned only with which feature structures satisfy it. By contrast, a formula is called a **constraint** when we are concerned with which feature structures model it.

A **grammar** is just a set of formulas viewed as constraints. If G is a grammar and $\mathcal{G}$ a feature structure which models G, we also say G **generates** $\mathcal{G}$, or $\mathcal{G}$ is **well-formed** relative to G. [3] We can now consider some examples of constraints typically employed in HPSG grammars.

[3] Strictly speaking, the grammar generates only the models which are canonical representatives of isomorphism classes of models. See Pollard 1998.

2.11 Species disjointness

It is standard to assume that, in order to be well-formed, a feature structure must be such that each of its nodes belongs to only one species. We can enforce this condition by including in the grammar, for each pair of distinct species names s and s' the constraint $\neg(s \wedge s')$.

2.12 Closed World Assumption

Another standard assumption in HPSG is that there aren't any linguistic entities beyond the species corresponding to the species names in $\mathcal{S}$. This can be thought of as a version of the fragment methodology. Another way to say this is that in order for a feature structure to be well-formed, every one of its nodes must belong to at least one (and given species disjointness, therefore exactly one) of the species. This can be expressed as the constraint:

$$\bigvee_{i=1}^{n} s_i \text{ (where } \mathcal{S} = \{s_1, \cdots, s_n\})$$

(Note: for $\phi_1, \cdots, \phi_n$ formulas, the notation $\bigvee_{i=1}^{n} \phi_i$ is a shorthand interpreted as follows:

for n=1, $\bigvee_{i=1}^{n} \phi_i$ means ϕ_1

for n > 1, it means $(\bigvee_{i=1}^{n-1} \phi_i) \vee \phi_n$

Thus, e.g., if n $= 3$, $\bigvee_{i=1}^{n} \phi_i$ is shorthand for $((\phi_1 \vee \phi_2) \vee \phi_3)$. A similar convention applies for $\bigwedge_{i=1}^{n} \phi_i$.)

2.13 Feature Geometry Constraints

These are constraints which tell us what features different species have, and what species the values of those features belong to. In HPSG it is standardly assumed that for each feature, there is a set of species such that that feature is defined for all nodes belonging to that species, but undefined for all other nodes. Thus, for each feature *f*, there is a constraint of the form

$$((f \doteq f) \leftrightarrow \bigvee_{i=1}^{m} t_i)$$

where each of the t_i is a species. For example, in HPSG, words and phrases (but nothing else) have the feature PHONOLOGY. Thus HPSG grammars contain the constraint[4]

$$((\text{ PHONOLOGY} \doteq \text{PHONOLOGY}) \leftrightarrow (word \vee phrase))$$

(Thus, anything that has a PHONOLOGY attribute is either a word or a phrase, and every word or phrase has a PHONOLOGY attribute.)

Another type of feature geometry constraint tells, for a given species and a given feature defined for that species, what species the value of that feature can be. Such constraints have the form

[4]As in propositional logic, "$\phi \leftrightarrow \psi$" is shorthand for $((\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi))$.

$$(s \rightarrow (f : \bigvee_{i=1}^{m} t_i))$$

For example, a German HPSG grammar might include the constraint

$$(noun \rightarrow (\text{CASE} : (nom \vee acc \vee gen \vee dat))).$$

2.14 Other constraints

Besides the feature geometry constraints, species disjointness, and the closed world assumption (which are sometimes referred to collectively as *ontological* constraints), grammars must contain further constraints which serve to determine (among those feature structures which satisfy the ontological constraints) which feature structures are well-formed. These constraints do the work done in other frameworks by such devices as lexical entries, phrase-structure (or immediate-dominance) rules, and principles of well-formedness (e.g., Subjacency, Binding Principle A, etc.). For example, it is standardly assumed in HPSG that for all headed phrases, the "head features" are the same as on the head daughter. This is formally expressed by the constraint

$$(\textit{headed-phrase} \rightarrow (\text{SYNSEM}|\text{LOC}|\text{CAT}|\text{HEAD} \doteq \\ \text{HEAD-DTR}|\text{SYNSEM}|XS\text{LOC}|\text{CAT}|\text{HEAD}\ ))$$

In HPSG, the lexicon is also a constraint. It has the form

$$(word \rightarrow \bigvee_{i=1}^{N} \lambda_i)$$

where each of the λ_i is a lexical entry. That is, each lexical entry is a formula (or description) which presents one of the (finitely many) options for a feature structure of species *word* to be a well-formed (structural representation of a) word. The lexicon, as a constraint, simply requires that every word take one of these options.

Similarly, there is a constraint on phrases of the form

$$(phrase \rightarrow \bigvee_{i=1}^{M} \rho_i)$$

where each of the ρ_i is an immediate dominance schema (or phrase-type). That is, each of the ρ_i presents one of the (small number of) options for a feature structure of species *phrase* to be a well-formed (structural representation of a) phrase. Thus the ρ_i do the same kind of work in HPSG as schematic immediate dominance rules (such as the rules of $\bar{\text{X}}$ theory) do in GB theory.

2.15 Formal properties of King formulas and grammars

Here we assemble some basic facts about formal properties of King formulas and grammars. For this discussion, assume we are given a set $\mathcal{O}$ of ontological constraints as discussed in 11-13 above. A feature structure is called **ontologically acceptable** if it satisfies all the ontological constraints (species disjointness, closed world, and feature geometry).

a. There is a proof theory for $\mathcal{K}$ which is sound and complete relative to ontological acceptability. Among other things, this means that for any two formulas ϕ and ψ, $\mathcal{O} \vdash (\phi \rightarrow \psi)$ iff every ontologically acceptable feature structure that satisfies ϕ also satisfies ψ.

b. Given a formula ϕ, it is decidable whether or not ϕ is satisfied by some ontologically acceptable feature structure or other. Likewise (since negation is classical), it is decidable whether ϕ is valid relative to ontological acceptability (i.e., whether EVERY ontologically acceptable feature structure satisfies ϕ).

c. Given a grammar G and a formula ϕ it is in general UNdecidable whether G predicts a feature structure which satisfies ϕ (or equivalently, whether there is a feature structure which models G and satisfies ϕ).

d. Given a set of ontological constraints $\mathcal{O}$ and a formula ϕ, there is a formula ϕ' which is (semantically) equivalent to ϕ relative to $\mathcal{O}$ (i.e., ϕ and ϕ' are satisfied by the same ontologically acceptable feature structures) such that ϕ' is a disjunction of conjunctions of formulas of one of the three forms $(\pi : s)$, $(\pi_1 \doteq \pi_2)$, and $(\pi_1 \not\doteq \pi_2)$. Thus, when $\mathcal{K}$ is being used only for description (as opposed to for constraints), no descriptive power is sacrificed by abstaining from the use of $\neg$ and $\rightarrow$.

2.16 Attribute-value matrices (AVMs)

King formulas are useful when great formal precision is needed (e.g., in proofs or in designing computer implementations). But they are too long and too hard to understand quickly for everyday linguistic analysis. For this reason, HPSG linguists usually use a different kind of description, attribute-value matrices (AVMs), in normal practice. The general format of an AVM is as follows:

a. a **tag** (boxed numeral like $\boxed{7}$) is an AVM.

b. a species symbol (e.g., *word*) or a disjunction of species symbols (e.g., *word* $\vee$ *phrase*) is an AVM; this kind of AVM is called a **sort description**. (Cf. section 19 below.)

c. a **bracketed description**, including the empty description '[]', is an AVM.

This has the form $\begin{bmatrix} \text{PATH}_1 & AVM_1 \\ \vdots & \\ \text{PATH}_n & AVM_n \end{bmatrix}$

In any AVM, any non-null combination of (a), (b), and (c) can be present. Thus, the following possibilities are available:

a. just a tag: $\boxed{n}$;

b. just a sort description: s, or $s_1 \vee \cdots \vee s_n$;

c. just a bracketed description;

d. both (a) and (b), e.g., $\boxed{n}s$;

e. both (a) and (c), e.g., $\boxed{n}\begin{bmatrix} \cdots \\ \cdots \end{bmatrix}$

f. both (b) and (c), e.g.,

$$\begin{bmatrix} s \\ \ldots \\ \ldots \end{bmatrix}$$

g. all three, e.g.,

$$\boxed{n}\begin{bmatrix} s \\ \ldots \\ \ldots \end{bmatrix}$$

Note that tags go outside the left bracket, and sort specifications go inside the left bracket at the top (or in some work, outside the left bracket at the bottom). Sort descriptions are equivalent to the identical King formulas. The lines inside a bracketed description are interpreted as equivalent to CONJUNCTION of King formulas. Co-occurrence of two or more instances of the same tag in an AVM is equivalent to path equalities ($\pi_1 \doteq \pi_2$). For example, the King formula

$$\begin{gathered}((\text{SUBJ}|\text{HEAD}:((\text{CASE: } \textit{nom}) \wedge (\text{AGR } ((\text{PER: } \textit{3rd}) \wedge (\text{NUM: } \textit{sing}))))) \wedge \\ (\text{HEAD: } (\textit{verb} \wedge (\text{VFORM: } \textit{fin}) \wedge (\text{AUX: } -))) \wedge \\ (\text{SUBJ}|\text{HEAD}|\text{AGR} \doteq \text{HEAD}|\text{AGR}))\end{gathered}$$

is equivalent to the AVM

$$\begin{bmatrix} \text{SUBJ}|\text{HEAD} & \begin{bmatrix} \text{CASE} & \textit{nom} \\ \text{AGR} & \boxed{1}\begin{bmatrix} \text{PER} & \textit{3rd} \\ \text{NUM} & \textit{sing} \end{bmatrix} \end{bmatrix} \\ \text{HEAD} & \begin{bmatrix} \textit{verb} & \\ \text{VFORM} & \textit{fin} \\ \text{AUX} & - \\ \text{AGR} & \boxed{1} \end{bmatrix} \end{bmatrix}$$

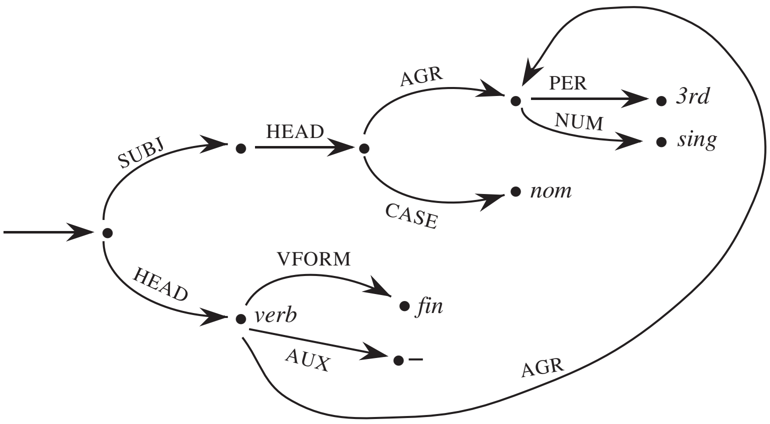
As a matter of style, when several paths share a prefix, that prefix is normally written only once for all the paths. Thus, we write:

$$\begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{VFORM} & \textit{fin} \\ \text{AUX} & - \end{bmatrix} \end{bmatrix}$$

rather than:

$$\begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{VFORM} & \textit{fin} \end{bmatrix} \\ \text{HEAD} & \begin{bmatrix} \text{AUX} & - \end{bmatrix} \end{bmatrix}$$

This example makes obvious the relative advantages and disadvantages of the two styles of description. Either one is satisfied by the following feature structure:



There is no standard equivalent of path INequalities for AVMs, but one obvious way to do it is to list path inequalities as additional lines inside a bracketed description. Thus, e.g., the King formula (SUBJ|INDEX $\neq$ OBJ|INDEX) would be expressed as the AVM:

$$\left[\textsc{subj}|\textsc{index} \neq \textsc{obj}|\textsc{index}\right]$$

A less obvious but more readable alternative is the following:

$$\begin{bmatrix} \textsc{subj}|\textsc{index} & \boxed{1} \\ \textsc{obj}|\textsc{index} & \boxed{2} \\ \boxed{1} \neq \boxed{2} & \end{bmatrix}$$

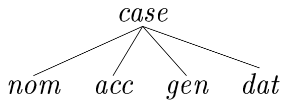
2.17 Subsumption and unification of AVMs

Given two AVMs μ_1 and μ_2, we say that μ_1 **subsumes** μ_2 if the class of feature structures that μ_1 describes is at least as big as the class of feature structures that μ_2 describes. If ϕ_1 and ϕ_2 are King formulas equivalent to μ_1 and μ_2 respectively, then μ_1 subsumes μ_2 iff ϕ_2 entails ϕ_1.

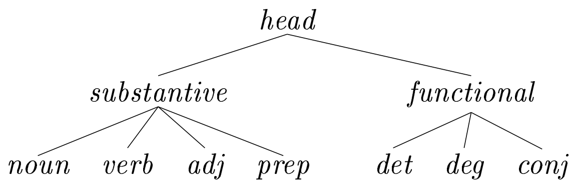
Again let μ_1 and μ_2 be two AVMs, with equivalent King formulas ϕ_1 and ϕ_2 respectively. An AVM μ_3 is called the **unification** of μ_1 and μ_2 (or more correctly, **a unification** of μ_1 and μ_2) if μ_3 is equivalent to $\phi_1 \wedge \phi_2$. Thus, μ_3 will describe those feature structures which both μ_1 and μ_2 describe (and no others).

2.18 Sorts and sort hierarchies

In writing HPSG grammars, certain disjunctions of species names are used repeatedly. For example, in a German grammar, one might often have occasion to employ the disjunction ($nom \vee acc \vee gen \vee dat$) (as in the constraint at the end of Sec. 13). Similarly, the disjunction ($word \vee phrase$) recurs frequently. Frequently used disjunctions are usually abbreviated with a single symbol, so that ($word \vee phrase$) is usually abbreviated *sign* and ($nom \vee acc \vee gen \vee dat$) is usually abbreviated *case*. Such abbreviations make grammars much easier to read and write (e.g., ((PHONOLOGY $\doteq$ PHONOLOGY) $\leftrightarrow$ (*sign*)), (*noun* $\rightarrow$ (CASE: *case*))). Of course, such abbreviations are meaningless unless they are made explicit. This is typically done by means of a diagram called a *sort hierarchy* (where a sort is just a symbol that is either a species name or an abbreviation for a disjunction of species names). Thus, the sort hierarchy



is equivalent to defining the symbol "*case*" as an abbreviation for "($nom \vee acc \vee gen \vee dat$)." More generally, sorts can be used as abbreviations for disjunctions of sorts (each of which in turn may be either a species name or an abbreviation). Thus, for example, the sort *head* may be used to abbreviate the disjunction of the parts of speech *noun, verb, adj, prep, det, conj, deg* (for the sake of this example, we assume that these are all the parts of speech). But among the parts of speech we might want to group the first four together as *substantive* and the last three as *functional.* (Typically we group species or sorts together when there is some feature they have in common, or there is some constraint that applies to just them.) We can do this by positing the sort hierarchy



Given two sorts σ_1 and σ_2 we say σ_1 is a subsort of σ_2 if it is more specific than σ_2 in the sort hierarchy. Thus, e.g., *prep* and *functional* are subsorts of *head.* Alternatively, if σ_1 is a subsort of σ_2, we can also say σ_2 is a supersort of σ_1.

Note: for any sort hierarchy, it is always assumed that the subsorts are pairwise incompatible (mutually exclusive). Thus, positing a hierarchy amounts logically to adding to the grammar a set of constraints of the form:

$$((\sigma \leftrightarrow (\sigma_1 \vee \cdots \vee \sigma_n))$$

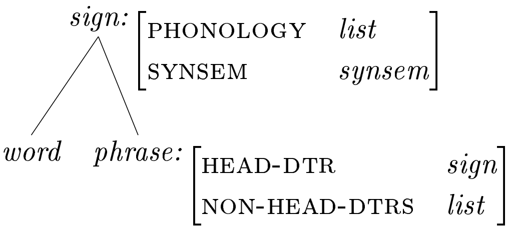
and

$$\neg(\sigma_i \wedge \sigma_j) \text{ (for all i, j = 1, ... n, with i} \neq \text{j)}$$

2.19 Feature declarations

Feature declarations, employed in conjunction with sort hierarchies, are a convenient notational alternative to feature geometry constraints; feature declarations together with sort hierarchies are the ususal way to express ontological constraints informally.

For example, the constraint ($noun \rightarrow$ (CASE : ($nom \vee acc \vee gen \vee dat$))) is expressed as the feature declaration *noun*: [CASE *case*], where the sort hierarchy headed by *case* is defined as above. To take a more complex example, the sort hierarchy with feature declarations:



is equivalent to the following set of feature geometry constraints (assuming that *word*, *phrase*, and *synsem*, and *list* are species):

$((word \vee phrase) \leftrightarrow ((\text{PHONOLOGY} \doteq \text{PHONOLOGY}) \wedge (\text{SYNSEM} \doteq \text{SYNSEM})))$

$(phrase \leftrightarrow (\text{DTRS} \doteq \text{DTRS}))$

$((word \lor phrase) \to ((\text{PHONOLOGY: } list) \land (\text{SYNSEM: } synsem)))$

$(phrase \to ((\text{HEAD-DTR: } word \lor phrase) \land (\text{NON-HEAD-DTRS: } list)))$

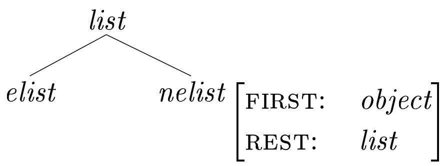
2.20 Lists

In HPSG, lists are usually introduced in the following way. We assume species names *elist* (empty list) and *nelist* (nonempty list), and feature names FIRST and REST, subject to the following feature geometry constraints:

$(nelist \leftrightarrow ((\text{FIRST} \doteq \text{FIRST}) \land (\text{REST} \doteq \text{REST})))$

$(nelist \to (\text{REST: } (elist \lor nelist)))$

Informally, we can say the same thing with a sort hierarchy and feature declarations, as follows:



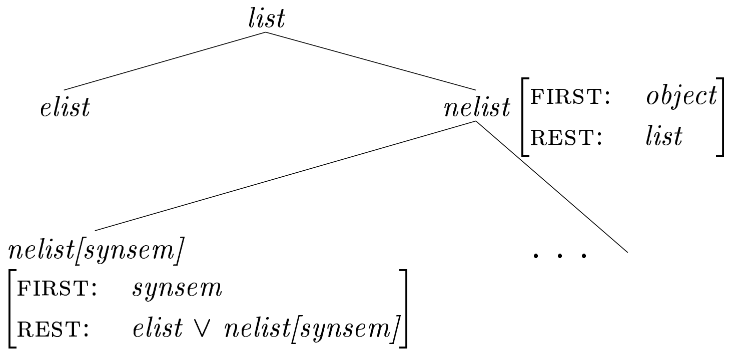
where *object* is a sort that abbreviates the disjunction of all the species names. (Thus if all the sorts are arranged in a single hierarchy, *object* is at the top.) The intuition is just that a list can be either an empty list or a nonempty list, and in the latter case it has a FIRST, which can be anything, and a REST, which is a list.[5] In AVMs, list descriptions are usually abbreviated by angle-bracket notation. Thus, $<a,\ b,\ c>$ abbreviates the AVM:

$$\begin{bmatrix} \textit{nelist} & \\ \text{FIRST} & a \\ \text{REST} & \begin{bmatrix} \textit{nelist} & \\ \text{FIRST} & b \\ \text{REST} & \begin{bmatrix} \textit{nelist} & \\ \text{FIRST} & c \\ \text{REST} & \textit{elist} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Also, in AVMs, *elist* is often written $<>$.

In practice, we usually want to work with lists all of whose members are of the same sort. In order to do this, we would need to revise the sort hierachy to contain feature declarations along the following lines, with a new species of nonempty list $nelist[\sigma]$ for each species (or sort) of thing σ that we need lists of, as follows:

[5]The LISP notions LIST, NIL, CONS, CAR, and CDR correspond to *list, elist, nelist*, FIRST and REST, respectively.



2.21 A new kind of formula: path relations

For each natural number n, we introduce a finite set of *n-ary relation names*. If R is an n-ary relation name and $\pi_1, \cdots, \pi_n$ are paths, then $\mathrm{R}(\pi_1, \cdots, \pi_n)$ is a formula. Of course, to be useful, we must be told what it means for a feature structure to satisfy a path relation formula. There are a number of different proposals about how to do this (e.g. Richter in preparation), not all of which have the same consequences.

Append

Append is a ternary relation symbol. Path relation formulas employing *Append* are much-used in HPSG since they are used to express that one list is the concatenation of two other lists. More precisely, satisfaction for *Append* formulas is defined in such a way that, for any three paths π_1, π_2, π_3, [6]

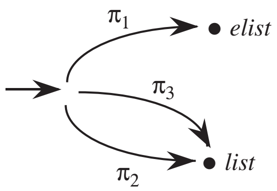
$\mathcal{G} \models \mathit{Append}(\pi_1, \pi_2, \pi_3)$ iff

1. $\mathcal{G} \models ((\pi_1 : \mathit{list}) \wedge (\pi_2 : \mathit{list}) \wedge (\pi_3 : \mathit{list}))$ and
2. either

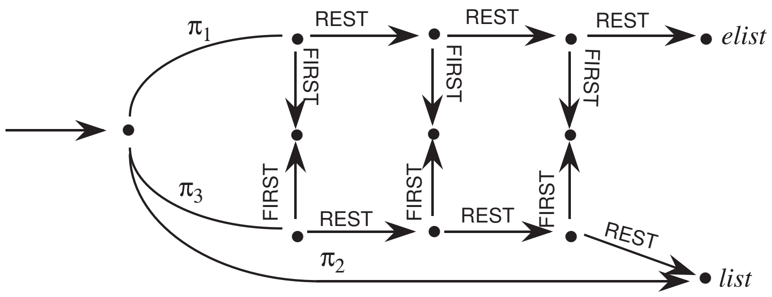
   a. $\mathcal{G} \models ((\pi_1 : \mathit{elist}) \wedge (\pi_2 \doteq \pi_3))$ or

   b. $\mathcal{G} \models (\pi_1|\textsc{first} \doteq \pi_3|\textsc{first})$ and $\mathcal{G} \models \mathit{Append}(\pi_1|\textsc{rest}, \pi_2, \pi_3|\textsc{rest})$

The two options are illustrated by the following two schematic graphs:



[6] The following biconditional statement does not actually define what it means for a feature structure to satisfy an *Append*-formula, but it places a very strong constraint on possible definitions.



Append-SYNSEMs

The ternary relation symbol *Append*-SYNSEMS is employed in an important constraint called the Valence Principle. Intuitively speaking, satisfaction for *Append*-SYNSEMS formulas is defined in such a way that $\mathcal{G} \models Append\text{-}\textsc{synsems}(\pi_1, \pi_2, \pi_3)$ if π_1, π_2 and π_3 are all lists and π_3 is the concatenation of the list of SYNSEM values of the π_1 list with the π_2 list. More precisely, it is defined in such a way that the following biconditional holds:

$\mathcal{G} \models Append\text{-}\textsc{synsems}(\pi_1, \pi_2, \pi_3)$ iff

1. $\mathcal{G} \models ((\pi_1 : list) \wedge (\pi_2 : list) \wedge (\pi_3 : list))$ and
2. either
   a. $\mathcal{G} \models ((\pi_1 : elist) \wedge (\pi_2 \doteq \pi_3))$ or
   b. $\mathcal{G} \models ((\pi_1 : nelist) \wedge (\pi_1|\textsc{first}|\textsc{synsem} = \pi_3|\textsc{first}) \wedge Append\text{-}\textsc{synsems}(\pi_1|\textsc{rest}, \pi_2, \pi_3|\textsc{rest}))$

Schematically:

$$\begin{bmatrix} \pi_1 & < \begin{bmatrix}\textsc{synsem}\ \boxed{1}\end{bmatrix}, \begin{bmatrix}\textsc{synsem}\ \boxed{2}\end{bmatrix}, \begin{bmatrix}\textsc{synsem}\ \boxed{3}\end{bmatrix} > \\ \pi_2 & \boxed{4} \\ \pi_3 & < \boxed{1}, \boxed{2}, \boxed{3}|\boxed{4} > \end{bmatrix}$$

The Valence Principle is the following constraint:

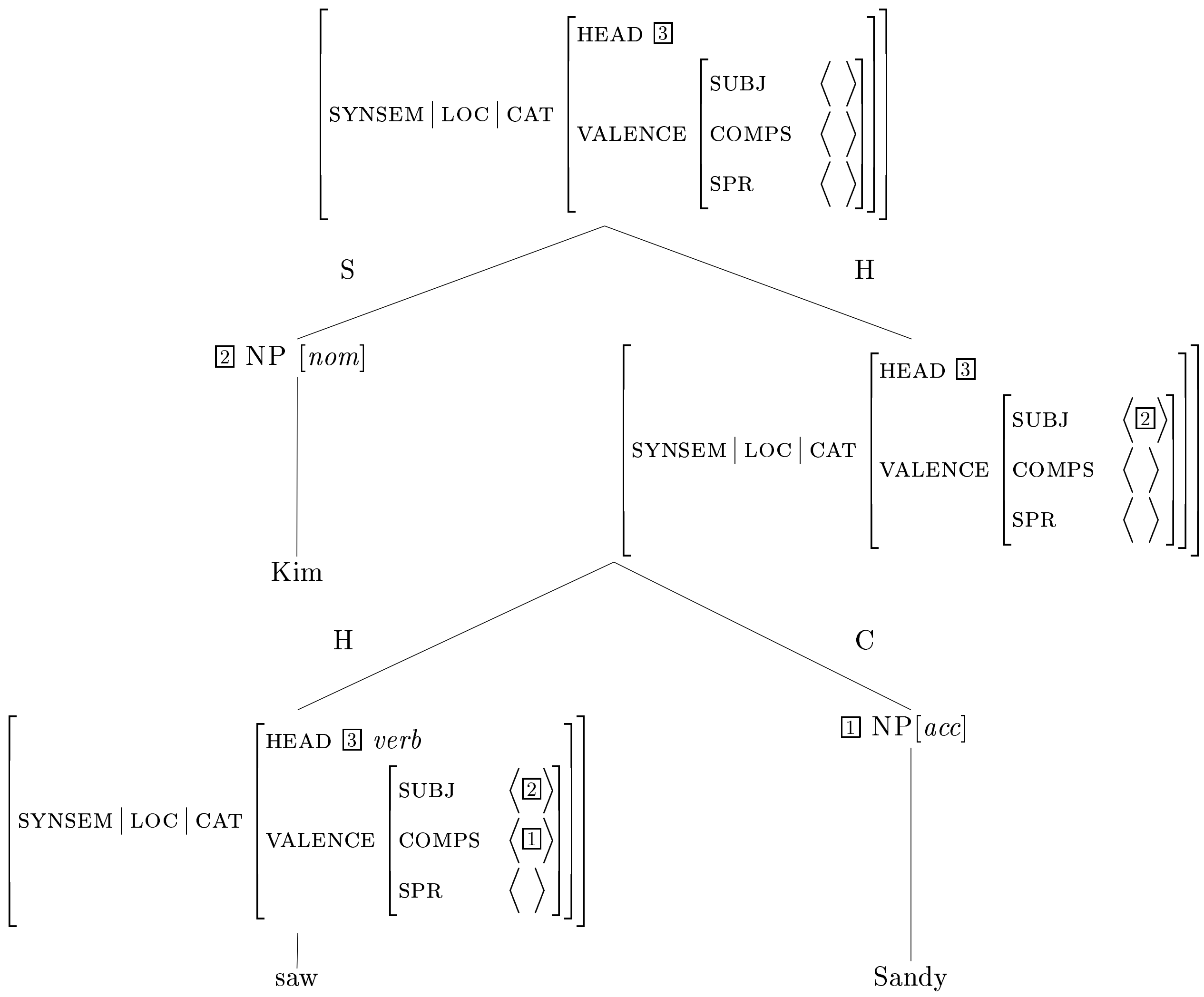
$$headed\text{-}phrase \rightarrow \begin{bmatrix} \textsc{synsem}|\textsc{loc}|\textsc{cat}|\textsc{val} \begin{bmatrix} \textsc{subj} & \boxed{4} \\ \textsc{comps} & \boxed{5} \\ \textsc{spr} & \boxed{6} \end{bmatrix} \\ \textsc{head-dtr}|\textsc{synsem}|\textsc{loc}|\textsc{cat}|\textsc{val} \begin{bmatrix} \textsc{subj} & \boxed{7} \\ \textsc{comps} & \boxed{8} \\ \textsc{spr} & \boxed{9} \end{bmatrix} \\ \textsc{subj-dtr}\ \ \boxed{1} \\ \textsc{comp-dtrs}\ \ \boxed{2} \\ \textsc{spr-dtr}\ \ \boxed{3} \\ Append\text{-}\textsc{synsems}(\boxed{1}, \boxed{4}, \boxed{7}) \\ Append\text{-}\textsc{synsems}(\boxed{2}, \boxed{5}, \boxed{8}) \\ Append\text{-}\textsc{sc synsems}(\boxed{3}, \boxed{6}, \boxed{9}) \end{bmatrix}$$

This says that, in a headed phrase, for each valence feature F, phrase, the F value of the head deaughter is the concatenation of the list of SYNSEM values of the F-daughters with the F-value of the phrase itself.

An alternative "functional" notation for AVMs employing relations is illustrated in the following expression of the Valence Principle"

$$\begin{bmatrix} \text{SYNSEM}|\text{LOC}|\text{CAT}|\text{VAL} & \begin{bmatrix} \text{SUBJ} & \boxed{4} \\ \text{COMPS} & \boxed{5} \\ \text{SPR} & \boxed{6} \end{bmatrix} \\ \text{HEAD-DTR}|\text{SYNSEM}|\text{LOC}|\text{CAT}|\text{VAL} & \begin{bmatrix} \text{SUBJ} & \mathit{Append}\text{-SYNSEMS}(\boxed{1}, \boxed{4}) \\ \text{COMPS} & \mathit{Append}\text{-SYNSEMS}(\boxed{2}, \boxed{5}) \\ \text{SPR} & \mathit{Append}\text{-SYNSEMS}(\boxed{3}, \boxed{6}) \end{bmatrix} \\ \text{SUBJ-DTR} & \boxed{1} \\ \text{COMP-DTRS} & \boxed{2} \\ \text{SPR-DTR} & \boxed{3} \end{bmatrix}$$

This following (conventionalized) description of a sentence illustrates the effect of the Valence Principle:


SYNSEM | LOC | CAT
HEAD 3
VALENCE
SUBJ ⟨ ⟩
COMPS ⟨ ⟩
SPR ⟨ ⟩
S
H
2 NP [nom]
Kim
SYNSEM | LOC | CAT
HEAD 3
VALENCE
SUBJ ⟨2⟩
COMPS ⟨ ⟩
SPR ⟨ ⟩
H
C
SYNSEM | LOC | CAT
HEAD 3 verb
VALENCE
SUBJ ⟨2⟩
COMPS ⟨1⟩
SPR ⟨ ⟩
saw
1 NP[acc]
Sandy

REFERENCES

Brody, Michael. 1995. Lexico-logical form. Linguistic Inquiry Monograph. Cambridge, MA.: MIT Press.

Jackendoff, Ray S. 1996. The architecture of the language faculty. Cambridge, MA: MIT Press.

King, Paul. 1989. A logical formalism for head-driven phrase-structure grammar. Manchester University Ph.D. Dissertation.

Koster, Jan. 1987 Domains and dynasties. Dordrecht: Foris.

Ladusaw, William. 1988. A proposed distinction between level and stratum. Linguistics in the morning calm 2, ed. by the Linguistic Society of Korea. Seoul: Hanshin Publishing Co.

Moshier, Drew. 1988. Extensions to unification grammars for the description of programming languages. University of Michigan Ph.D. dissertation.

Pullum, Geoffrey K. 1989. Formal linguistics meets the boojum. Natural Language and Linguistic Theory 7: 137-143.

Richter, Frank. In preparation. Eine formale Sprache für HPSG und ihre Anwendung in einem Syntaxfragment des Deutschen. Ph.D. thesis, Eberhard-Karls-Universität, Tübingen.