# An Informal Sketch of a Formal Architecture for Construction Grammar

Paul Kay
UC Berkeley

July 16, 1998

**Abstract**

A formal architecture for Construction Grammar (CG) is sketched. Modeling domain objects (constructs) are constituent structures with feature structures at the nodes, aka Feature Structure Trees (FTs). Constructions are partial descriptions of FTs, expressed as sets of constituent structure equations and path equations. Unification of constructions is defined. A construct $c$ is licensed by a grammar $G$ iff $G$ contains a set of constructions whose unification fully describes $c$.

## 1 Introductory

The major empirical motivation for Construction Grammar (CG) is the need to develop a system of grammatical description in which the marked constructions (more or less 'idiomlike' forms of expression) are represented in the same formal system as the regular, 'core' patterns or rules (Fillmore & Kay Forthcoming, Kay & Fillmore In Press).[1] The original representation system for Construction Grammar (CG) was a system that represents constructions (items of grammar) and constructs (items of language: sentences, phrases and words) in 'box diagrams', which depict constituent structure as nested boxes, with the syntactic, semantic, lexical and phonological information associated with each constituent written in the box representing that constituent.

In the example in (1) below, a highly oversimplified representation of the sentence *Sid sat*, the outer box represents the sentence as a constituent, the small, left-daughter box indicating the subject (*Sid*) constituent and the right-daughter box representing the predicate (*sat*) constituent. The feature structure enclosed only in the larger box, at the top, says that (1) syntactically we have a sentence, (2) semantically, (a) the frame (relation) is SITTING,

[1]Charles Fillmore originated the approach to a construction-based grammar for which a formal architecture is sketeched here. It is a pleasure to acknowledge the contributions to this paper of Ron Kaplan and Mary Dalrymple, without whose stimulation and advice it would not have been developed. Charles Fillmore, Andreas Kathol, and J.-P. Koenig, and participants in a seminar at Stanford University in Spring 1997 have provided valuable suggestions. Versions of this material have been presented at Stanford University, the University of Konstanz, the Max Planck Institute in Nijmegen, and the 1998 Conference on Formal, Head-Driven, and Categorial Grammar in Saarbrücken. Errors are the author's.

(b) the unique argument is SIDNEY and (c) the tense is past, and (3) the phonology of the sentence is schematically represented as <sɪdsæt>. The entries in each of the smaller, daughter, boxes can be read in the same way.

(1)

| |
|---|
| syn S<br>sem [frame SITTING<br>args {SIDNEY}<br>tense past]<br>phon <sɪdsæt><br><br>[syn NP<br>sem SIDNEY<br>phon <sɪd>]<br><br>[syn VP<br>sem [frame SITTING<br>args {SIDNEY}<br>tense past]<br>phon <sæt>] |

The above example is of the *construct* (piece of English) *Sid sat.* The grammar of English **licenses** this **construct** by **unifying** or **combining** certain **constructions**. Constructions are also represented by box diagrams and the unification of the constructions licensing a construct was described as 'overlaying' or 'combining' constructions so that the parts they share match up and the parts they don't share are added to the whole.

This paper takes a first step toward specifying what kind of formal objects these box diagrams are pictures of and what is really meant by 'matching up', 'combining' or 'unifying' two box diagrams, each of which gives a partial description of a construct (bit of language), into something that gives a fuller description of this construct, until finally we have licensed a sentential construct (i.e., a sentence). The paper concludes with a brief discussion of similarities and differences of CG with HPSG and LFG.

The empirical objects, sentences words and phrases, accounted for by a construction grammar are taken to have the form of constituent structure (CS) trees with feature structures (FSs) at the nodes. A construction is defined in terms of a set of simultaneous constraint equations, describing a class of such hybrid structures, rather like those of LFG. A hybrid structure which represents a fully parsed word, phrase or sentence of a language is called a construct of that language. Licensing a structure that satisfies more than one construction consists in putting together the relevant set of constructions (interpreted as a conjunction). The box notation of CG (Fillmore & Kay Forthcoming; Kay & Fillmore In Press) is preserved by translating back and forth between that notation and an algebraic notation that looks much like LFG functional equations.

We first define FSs, then feature structure trees (FTs). The latter are the 'hybrid' structures: CS trees with FSs at the nodes. Modeling-domain objects (constructs) are FTs. CG modeling-domain objects are, thus, somewhat like those of LFG except that, like HPSG, phonological, semantic, categorial and 'bar' information are included in the FSs: no separate semantic and phonological modules and no labeling function, like LFG's $\lambda$, are posited. There is nothing corresponding to the up and down arrows of LFG. Also, LFG emhasizes the separateness of C-structures, F-structures and rules of correspondence between them, while CG emphasizes the unity of the construction, which, as in HPSG types (Pollard & Sag

1987; Pollard & Sag 1994, and especially Sag 1997), combines CS and non-CS aspects in a single unit.

Constructions are descriptions of classes of FTs, expressed as sets of constraint equations. Some of these equations express the constituent structure, as relations of dominance and precedence between nodes. Other equations are path equations which identify the CS node to which the beginning FS of each path is anchored. So we have equations that say things like "(The FS at node-1)'s syntax's head's cat is v." (Cf. Kaplan & Bresnan 1982:182ff, Kaplan 1995.)

# 2 Primitive Notions

- attributes
- atomic values
- phonological representations
- (constituent structure) nodes

# 3 Feature Structures

(2) Let the expression $(D \rightarrow R)$ denote the set of partial functions from domain $D$ to range $R$.

The primitive notions of the current exposition are attributes, atomic values, phonological representations, and (constituent structure) nodes.

(3) $A$: a finite set of attributes,
$V$: a finite set of atomic values,
$P$: a denumerable set of phonological representations.

A function $F$ is a feature structure (FS) iff $F \in (A \rightarrow V \cup P \cup F \cup \wp(F))$.[2]

A feature structure is a partial function from a finite set of attributes to a range consisting of objects of any of the following four types: atomic values, phonological representations, feature structures, sets of feature structures.

# 4 Feature Structure Trees (FTs)

(4) The symbol '$<$' is used to denote the notion 'precedes'.

(5) $N$: a finite set of CS nodes,
$F$: a finite set of FSs,
$M$(other): a function in $(N \rightarrow N)$;
$<$ ('precedes'): a strict partial order on $N$;

[2] '$\wp(F)$' denotes the power set of $F$.

$\phi$: a function in $(N \rightarrow F)$.

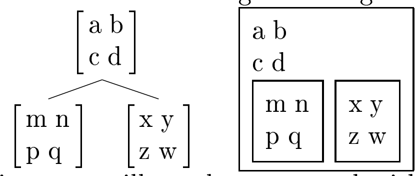
A five-tuple $t = < N, <, F, M, \phi >$ is a *Feature Structure Tree* (FT) iff
(i) Exactly one node $r \in N$ has no mother in N,
(ii) Every node $n \in N$ other than $r$ has exactly one mother,
(iii) For distinct nodes $n_i, n_j \in N$, if $n_i$ and $n_j$ are sisters, then $(n_i < n_j)$ or $(n_j < n_i)$,
(iv) For $n_i, n_j \in N$, if $M(n_i) < M(n_j)$, then $n_i < n_j$.

FTs are the formal objects which represent constructs. Note that no variables exist in the space of FTs and all precedence relations are specified by conditions (iii) and (iv).

# 5 Descriptions (Equations)

Let's start with a construct expressed in CG box notation and see what a description of it will look like in equation form.[3]

[3]CG often uses nested boxes rather than a structure of branching nodes to represent constituent structure. Thus, the two diagrams below are notational variants for a single FT. In the present paper, constructs are represented in branching node diagrams and constructions in box diagrams:



This paper will not be concerned with the feature geometry of any particular implementation of the CG approach. In order to simplify each example as much as possible, no attempt has been made to have all examples imply a single, consistent feature geometry. This paper is concerned only with the formal architecture in which CG grammars are expressed, not with the particulars of any one grammar or with a general theory of CG grammars. Also, to simplify some diagrams, paths are abbreviated to their last one or two attributes (e.g., diagram (35)), since the full path is in principle recoverable from knowledge of the feature geometry. One version of a CG feature geometry which seems to work fairly well for English is summed up in the following diagram.

$$\boxed{\begin{array}{ll}
\text{role } [\,] & \\
\text{synsem} & \begin{bmatrix} \text{rel(ational)} & \begin{bmatrix} \text{g(rammatical) f(unction) } [\,] \\ \theta \text{ (semantic role)} \end{bmatrix} \\ & \begin{bmatrix} \text{syn} & \begin{bmatrix} \text{head} & \begin{bmatrix} \text{lexical-head } [\,] \\ \text{cat } [\,] \\ \ldots \end{bmatrix} \\ \text{level} & \begin{bmatrix} \text{max } [\,] \\ \text{lex } [\,] \\ \ldots \end{bmatrix} \end{bmatrix} \\ \text{sem } [\,] & \end{bmatrix} \end{bmatrix} \\
\text{val } \{\} & \\
\text{phon } <> &
\end{array}}$$

In CG notation, empty brackets, braces, etc., specify, not the null set etc., but a variable of the appropriate type. For example, { } denotes an arbitrary set, [ ] an unspecified FS. Readers accustomed to HPSG usage should be wary.

(6) *She sang.* (construct)

$$\begin{bmatrix} \text{sem} & \begin{bmatrix} \text{tense past} \\ \text{scenes } \{...\} \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \text{role comp} & \\ \text{syn} & \begin{bmatrix} \text{head} & \begin{bmatrix} \text{cat n} \\ \text{pro} + \end{bmatrix} \\ \text{level} & \begin{bmatrix} \text{max} + \\ \text{lex} + \end{bmatrix} \end{bmatrix} \\ \text{phon} & < \textit{she} > \end{bmatrix} \quad \begin{bmatrix} \text{role head} & \\ \text{syn} & \begin{bmatrix} \text{head} & \begin{bmatrix} \text{cat v} \\ \text{infl fin} \end{bmatrix} \\ \text{level} & \begin{bmatrix} \text{max} + \\ \text{lex} + \end{bmatrix} \end{bmatrix} \\ \text{sem} & \begin{bmatrix} \text{tense past} \\ \text{scenes } \{...\} \end{bmatrix} \\ \text{phon} & < \textit{sang} > \end{bmatrix}$$

(This representation of the construct is very partial, but we will pretend that it is complete.) We can write a description of (certain aspects of) the construct shown in (6) in the following way. First we define the constituent structure as consisting of a mother and two daughters.

## 5.1 CS equations

(7) a. $M(n_2) = n_1$[4] b. $M(n_3) = n_1$ c. $n_2 < n_3$

## 5.2 FS equations

Using the LFG type of notation for path equations, based on the idea that paths represent successive functional application, we can write, for example:

(8) $\phi(n_1)$ sem tense = past

Equation (8) can be read '(the feature structure assigned by $\phi$ to $n_1$)'s sem's tense is past.' We can now proceed to a full decription of *She sang* (to the degree that its structure is specified in (6)).

(9) a. $\phi(n_2)$ role = comp b. $\phi(n_2)$ syn head cat = n
c. $\phi(n_2)$ syn head pro = + d. $\phi(n_1)$ sem tense = $\phi(n_3)$ sem tense

Equation (9d) represents the kind of identity of information that is talked about in Fillmore & Kay Forthcoming as unification, and represented with unification variables that look like '#2'. When we come to the Subject-Predicate (S-P) construction, we will see that in the box diagram (35) this identity of information is represented by the occurrence of two instances of '#2[ ]'.

Although it is not shown in (6), we know that a fuller description of the construct *She sang* will also satisfy the following equation.

[4]More self-consciously: $\exists N, n_1, n_2[(n_1 \in N) \,\&\, (n_2 \in N) \,\&\, (M(n_2) = n_1)]$.

(10) $\phi(n_3)$ sem scenes $\subseteq \phi(n_1)$ sem scenes[5]

(Of course in this particular case, the more informative equation (11) also holds. But (10) is interesting because it will be true of every S-P construct.)

(11) $\phi(n_3)$ sem scenes $= \phi(n_1)$ sem scenes

A few more equations descriptive of *She sang*, just to give a little more of the flavor.

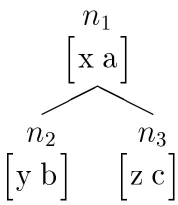
(12) a. $\phi(n_2)$ phon $= <$ *she* $>$ b. $\phi(n_3)$ role $=$ head
c. $\phi(n_3)$ syn head cat $=$ v d. $\phi(n_3)$ phon $= <$ *sang* $>$

We pretend equations (7–12) fully characterize the construct *She sang*. (There are two levels of pretense here (1) the pretense that diagram (6) fully characterizes the construct *She sang* and (2) the pretense that equations (7–12) fully characterize diagram (6).[6]) Any proper subset $p$ of this set of equations will also describe (be satisfied by) *She sang* but will not describe *She sang* fully, in the sense that there will exist equations which are satisfied by *She sang* which are not in $p$ (or implied by $p$). For example, if $p$ is the set of equations (7–12c), $p$ describes *She sang* only partially because (12d) is satisfied by *She sang* and is not in $p$ (or implied by $p$).

## 6 Constructions and Licensing

In describing an FT, such as that shown in (13), we are forced, for bookkeeping purposes, to put numerical labels (or something equivalent) at the nodes, so as to be able to identify individual nodes in stating constraint equations such as those in (14–19), but if, for example, we had stated (13) as (20), yielding the equations in (21–26), we would have given a description of a structure which we would like to consider identical to the structure in (13).

(13)




(14) $\phi(n_1)x = a$

(15) $\phi(n_2)y = b$

[5]The scenes value in CG is roughly analogous to the LISZT value in HPSG minimal recursion semantics (MRS). Broadly, CG semantic representation is similar in spirit and form, though not precisely equivalent, to MRS. Substantively, (10) means roughly that the semantics of a sentence will include everything in the semantics of its verb phrase.

[6]One respect in which equations (7–12) do not fully characterize the structure in diagram (6) is that these equations don't tell us that $n_2$ and $n_3$ are the only daughters of $n_1$. We could, for example, define predicates for the description language that would allow us to specify $n_2$ as the leftmost daughter of $n_1$ and $n_3$ as the rightmost daughter of $n_1$. More generally, we allow the description language to contain elements that are not part of the direct representation of modeling domain objects, e.g., negation, disjunction and material implication.

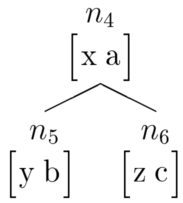
(16) $\phi(n_3)z = c$

(17) $M(n_2) = n_1$

(18) $M(n_3) = n_1$

(19) $n_2 < n_3$

(20)




(21) $\phi(n_4)x = a$

(22) $\phi(n_5)y = b$

(23) $\phi(n_6)z = c$

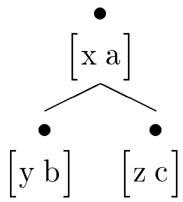
(24) $M(n_5) = n_4$

(25) $M(n_6) = n_4$

(26) $n_5 < n_6$

Actually, in writing either (13) or (20), the structure we are really after is adequately represented in (27).

(27)




If we look back to the definition of FTs in (5), we note that no mention is made of particular numerical labels for nodes. Numerical node labels were introduced only to facilitate the writing of descriptive equations. They are not an essential part of the model-domain structures. At some point we will need to free our characterizations of FTs from particular assignments of identifying numerals to nodes. Let us call sets of equations of the kind exemplified in (14–19) and (21–26) 'descriptions'.

(28) A *description* is a set of constraint equations which makes use of numerical labels to identify individual nodes.

We want to be able to say that technically distinct descriptions like those of (14–19) and (21–26) are mere 'renumberings' of each other.

(29) For two descriptions $D$, $D'$, we will say that $D'$ is a renumbering of $D$ iff there is a one-to-one function $g$ from node labels to node labels such that $g(D) = D'$ and $g^{-1}(D') = D$.

The definition in (29) will serve two purposes, one formal and one empirical. On the formal side, when we get around to combining two descriptions, call them $D_1$ and $D_2$, we will define their combination in terms of arbitrary renumberings of $D_1$ and $D_2$. This procedure will have the result that all logically possible identifications of a node in $D_1$ with a node in $D_2$ may appear in the combination (subject to whatever constraints are explicitly placed on such identifications in the definition of combination). Consequently, in combining descriptions we won't have to bother with renumbering nodes, making sure that $D_1$ and $D_2$ have disjoint sets of node labels to start out with, or anything of that sort. The empirical consequence of definition (29) is that it allows us to eliminate node labels from that part of the formal apparatus which is intended to be psychologically plausible. Numerical node labels are used by linguists in talking to each other, but these labels are not part of what the theory attributes to speakers. Speakers are held to combine, not descriptions—which contain node labels serving only to enable the analyst to keep the nodes straight—but arbitrary renumberings of descriptions.

(30) A description D is *satisfiable* iff there is at least one FT $t$ such that for each equation $d \in D$, $t$ satisfies d.

Any satisfiable description is a potential construction. The grammar of a language tells us which descriptions are the constructions of a given language. (The grammar also tells us the memberships of $A$, $V$, and (somehow) $P$.)

'Combining' constructions involves taking the union of the corresponding descriptions, subject to certain conditions to be described presently. Combining constructions produces sets of equations of increasing cardinality, providing increasingly detailed descriptions. We think of a sufficiently detailed description as licensing a particular FT. A description's licensing (or not licensing) an FT is defined in part in terms of the FT's satisfying (or not satisfying) the description. However, the relation between a licensing description and the FT it licenses must include, but not be restricted to, satisfaction of the former by the latter. A description may, as we have noted, only partially describe an FT which satisfies it.

Intuitively we want a licensing description to be a full, not merely a partial, description of the licensed FT: the description and the model-object (FT) it licenses should specify the 'same information'. That is, the licenser should contain no information not embodied in the model-object and the model-object should embody no information not contained in the licenser.

(31) A satisfiable description $D$ fully describes an FT $t$, iff
(i) $t$ satisfies each equation $d \in D$ and
(ii) for any description $D'$, if $t$ satisfies each equation $d' \in D'$, then $D \rightarrow D'$.

Property (i) ensures that $D$ contains no information not embodied in $t$. Property (ii) ensures that $t$ embodies no information not contained in $D$.[7] The relation defined in (31) is

[7]Suppose $t$ embodies all the information in $D$ plus that contained in a set of constraint equations $E$ (where $E$ is not implied by $D$). The description $D \cup E$ is then satisfied by $t$, violating (ii). So $D$ does not fully describe $t$.

sometimes described by calling the model object the 'minimal satisfier' of the description; the motivation for this terminology is that the model-object $t$ is the least detailed object which the description $D$ describes: if you added any more information to $t$, then $D$ would no longer describe $t$ (Kaplan & Bresnan 1982:201–203).[8] Our preferred terminology for the relation defined in (31) will be that $D$ 'licenses' $t$.

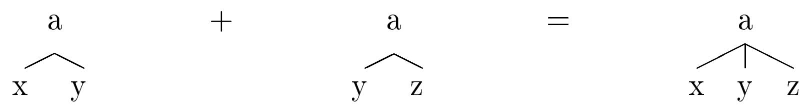
(32) For a description $D$ and an FT $t$, if $D$ fully describes $t$ we say equivalently that $D$ licenses $t$ or that $t$ is a minimal satisfier of $D$.

We want the grammar ($A$, $V$, $P$ and the constructions) to provide us with a description that licenses each construct in the language and with no description which licenses any FT that is not a construct of the language. This desideratum of a construction grammar corresponds to the familiar desideratum of generative grammar: a grammar of a language should account for all the sentences of the language, provide an analysis of the structure of each, and rule out all non-sentences.[9] Put slightly differently, the object of the game is to select $A$, $V$, $P$ and the constructions so that every FT which is a construct of the language is licensed by some combination of constructions and every combination of constructions that licenses an FT licenses a construct.

Having established what we want to do with combinations of descriptions, namely license constructs, we need to define combination of descriptions. A major part of the definition of combination of descriptions will involve forming unions of the sets of constraint equations that constitute the descriptions being combined.

In addition to forming the unions of the descriptions being combined, we will want to maintain the integrity of their sister sets (i.e., sets of all daughters of a single mother). For example, we do not want combination of descriptions to allow the sort of thing informally depicted in (33).

(33)




To permit the kind of combination illustrated in (33) would be tantamount to saying, for example, that if a grammar licenses constituents of the form [$_S$ NP VP] and [$_S$ VP NP], it automatically licenses constituents of the form [$_S$ NP VP NP] and [$_S$ VP NP VP]. Also, in combining descriptions we will need to avail ourselves of the freedom of node identification afforded by the notion of an arbitrary renumbering of a description (recall definition 29).

(34) A combination $C$ of two satisfiable descriptions $D$, $E$ is a satisfiable description $D' \cup E'$ such that
(i) $D'$ is a renumbering of $D$,
(ii) $E'$ is a renumbering of $E$, and

[8] This is the approach taken by Kaplan & Bresnan (1982:201–203) to the problem dealt with in (31ii). The Kaplan-Bresnan approach is to make sure that $t$ is the 'smallest' object that satisfies $D$ while the approach taken in (31ii) is to make sure that $D$ is the 'biggest' description that $t$ satisfies.

[9] On this usage, one does not speak of an 'ambiguous sentence' but of an 'ambiguous word string, which corresponds to more than one sentence'. Nothing important turns on this terminological point. Also, a construction grammar licenses words and phrases, as well as sentences.

(iii) the satisfaction set of C is the intersection of the satisfaction set of $D'$ and the satisfaction set of $E'$.

Clause (iii) of definition (34) rules out the unwanted situation illustrated in (33). Certain other things should be noted about definition (34). First, when the combination $C$ fully describes an FT, $D'$ and $E'$ will necessarily have at least one node in common. Secondly, the definition of renumbering (29) allows us to 'match up' nodes in D and nodes in $E$ in every possible way consistent with the dominance and precedence relations already specified in $D$ and $E$.[10] Thirdly, not every pair of satisfiable descriptions is combinable. The combined description must be such that there is at least one FT that satifies it. Moreover, there must be a way of assigning node match-ups such that condition (iii) of (34) is not violated. Fourthly, combination of descriptions is not an operation: there can be more than one combination of the same two descriptions.An example would be a case in which one description specifies something with the shape $[_{\mathrm{S}}$ NP $[_{\mathrm{VP}}$ V NP]] and the other description specifies something with the shape $[_{\mathrm{NP}}$ ]. The second description can describe either the subject or object NP of the first. Actually, the isolated NP could also be neither of these; it might, for example, be the subject of a higher clause of which the $[_{\mathrm{S}}$ NP $[_{\mathrm{VP}}$ V NP]] structure is a clausal complement. But if non-overlapping descriptions are combined 'initially', they will have to be connected 'eventually' if they are to contribute to a description that licenses a construct, which is an FT. The words 'initially' and 'eventually' are put in scare quotes because licensing is declarative: a licensing description can be 'put together' by combining all the constructions that go into the 'finished' description in any order. One can likewise think of parsing a construct as decomposing it into the constructions whose combination constitutes its licensing description.

## 6.1 Example: the Subject-Predicate (S-P) Construction

Some of the constraints (equations) in a construction will come from more general principles, which are simply more general constructions that are inherited by less specific constructions. An important subset of these constraints are of the (rough) form: If a construction specifies a daughter node to be 'role head' and it inherits the more abstract construction we're talking about, then the head daughter's ... x is related to the mother's ... x by the relation y (e.g., identity, subset).

In the diagram for a construction, we list at the top of the outer box in **boldface** the names of any constructions which are inherited by the one being diagrammed. For example, in the diagram of the Subject-Predicate construction in (35), it is indicated in this way that the Headed Phrase construction (41) is inherited. The notation in (35) **INHERIT Headed Phrase** does not denote an attribute-value pair. It may be thought of simply as a 'macro' which 'calls' the Headed Phrase construction to contribute all its information, i.e., all its equations, to the Subject-Predicate construction. More interestingly, The notation **INHERIT Headed Phrase** may be interpreted as expressing one link in an inheritance

[10] To prevent the licensing of non-constructs (overgeneration) we have to be sure to set up the constructions in a way that will prevent non-constructs from being licensed. But this is just the problem that every grammarian has in making sure the hypothesized rules don't overgenerate.

hierarchy of constructions which, in its entirety, represents the major grammatical generalizations of the language under study, for example, the generalization of English—and other languages—that lexical (and phrasal) heads share many syntactic and semantic properties with their phrasal mothers.

(35) Subject-Predicate (S-P) Construction (abbreviated)

$$\boxed{\begin{array}{ll}
\textbf{INHERIT Headed Phrase} & \\
\text{head cat [ ]} & \\
\text{sem} \begin{bmatrix} \text{tense \#2 [ ]} \\ \text{scenes \{\}} \end{bmatrix} & \\
\text{val \{\}} & \\
\boxed{\#1 \begin{bmatrix} \text{role comp} \\ \text{gf subj} \end{bmatrix}} & \boxed{\begin{array}{l} \text{role head} \\ \text{head cat v} \\ \text{max +} \\ \text{sem} \begin{bmatrix} \text{tense \#2 [ ]} \\ \text{scenes \{\}} \end{bmatrix} \\ \text{val \{\#1[ ]\}} \cup \text{\{ \}} \end{array}}
\end{array}}$$

Equations (36–40) express information presented diagrammatically in diagram (35). Equations (36) express constituent structure information about the construction. Equations (37) express path information which is peculiar to the S-P construction, specifically not inherited from the Headed Phrase (or any other construction). Equations (38–40) express information contained in the S-P construction that is inherited from the Headed Phrase construction, see (41).

(36) **CS Equations**

a. $M(n_2) = n_1$ b. $n_2 < n_3$ c. $M(n_3) < n_1$

(37) **Equations proper to S-P**[11]

a. $\phi(n_2) \in \phi(n_3)$ val b. $\phi(n_2)$ role = comp
c. $\phi(n_2)$ ... gf = subj d. $\phi(n_3)$ role = head
e. $\phi(n_3)$ ... cat = v
f. $\phi(n_1)$ ... sem tense = $\phi(n_3)$ ... sem tense

**FS Equations inherited from Headed Phrase. See (41)**

(38) $\phi(n_1)$ ... head cat = $\phi(n_3)$ ... head cat

(head feature percolation; '#2' in 41)

(39) $\phi(n_3)$ val $\subseteq \phi(n_1)$ val
(valence of head daughter is a subset of valence of mother; '#3' in 41)

(40) $\phi(n_3)$ ... sem scenes $\subseteq \phi(n_1)$ ... sem scenes
(scenes of head daughter are a subset of scenes of mother; '#1' in 41)

[11](37a) and (37f) correspond to '#1' and '#2' in (35), respectively.

## 7 Constructional Inheritance

The Headed Phrase (HP) construction, which is inherited by the S-P construction, among others, follows in diagram form.

(41) Headed Phrase (HP) (abstract) construction

$$\boxed{\begin{array}{l} \text{synsem intrinsic} \begin{bmatrix} \text{sem scenes } \#1\{\ \} \cup \{\ \} \\ \text{syn head } \#2[\ ] \end{bmatrix} \\ \text{val } \#3\{\ \} \cup \{\ \} \\ \boxed{\begin{array}{l} \text{role head} \\ \text{synsem intrinsic} \begin{bmatrix} \text{sem scenes} \#1\{\} \\ \text{syn head } \#2[\ ] \end{bmatrix} \\ \text{val } \#3\{\} \end{array}} , \boxed{\text{role comp}}^{+} \\ (\phi(n_i)\text{role} = \text{comp}) \rightarrow (\phi(n_i) \in (\phi(M(n_1))\text{val})) \end{array}}$$

The constraint expression at the bottom of diagram (41) says that the feature structure of each comp daughter unifies with (has the same structure as) a member of the valence value of the mother. There are cases, such as this, where it is more convenient to express a constraint directly in algebraic form than to create for it a special convention in the box diagram mode of representation.

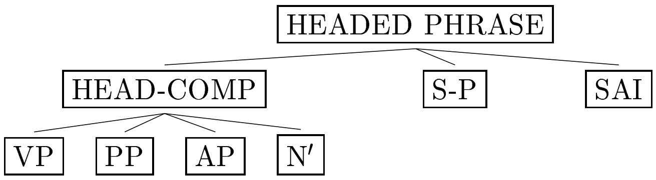
Another such case is illustrated in constraint expression (42).

(42) $(n_j = M(n_j)) \rightarrow (\phi(n_i)\text{phon} \in dom(\phi(n_i)\text{phon}))$

The constraint in (42) is part of every construction in which a mother and daughter(s) can be distinguished. It says that the phonology value of the mother node is a function of the phonology values of the daughter(s).[12]

All the properties of (41) are inherited by each of the more detailed constructions which license verb phrases, preposition phrases, adjective phrases, and 'n-bar' constituents, as well as the S-P construction, which licences subject-predicate clauses, and SAI, which licenses inverted clauses. The comma in (41) indicates that HP doesn't stipulate linear order between head and complement(s). The HEAD-COMPLEMENT construction, which inherits HP and is in turn inherited by VP, PP, AP and N-'bar', places the head before the complements.

(43)




(44) A construction $C_1$ *inherits* a distinct construction $C_0$ iff for some renumbering $C_0'$ of $C_0$, $C_0' \rightarrow C_1$.

We distinguish a 'compiled' grammar, within the full grammar, which contains only the 'ground level' constructions, that is, those constructions which have no heirs (the leaves of the inheritance hierarchy). This is the grammar that we propose is most relevant to actual

[12]Some details are omitted here for simplicity of exposition.

speakers' and hearers' 'on line' production and interpretation of utterances. If a description $D$ contains (implies) a construction $C_1$ and licenses a construct $t$, if there is a construction $C_0$ which $C_1$ inherits, then the description $D \cup C_0$ is equivalent to $D$ and will also license $t$. But since $D \cup C_0$ is not in the compiled grammar, we don't let this worry us.

## 7.1 Recursive licensing

Unlike LFG phrase structure rules and lexical items and unlike HPSG types, distinct constructions can span the same (piece of) FT. For example, the English VP construction, which provides for a lexical verb followed by an arbitrary number of constitutents (subject to valence restrictions), can unify with a construction specifically licensing a VP displaying the 'heavy NP shift' property. In order to specify an explicit recursive licensing procedure for sentences, we need some way to deal with this overlap of constructions. We wish to reduce the set of constructions of a grammar to a set of construction-like objects (let's call them CLOs) with the property that in licensing a given sentence, exactly one CLO licenses each node. To obtain the set of CLOs from the set of constructions $C$: (1) form the power set of the set of constructions $\wp(C)$; (2) for each set of constructions in $\wp(C)$, attempt to unify all the members, matching the root nodes; (3) throw away all the sets that don't unify; (4) the remainder is the set of CLOs.

# 8 CG, LFG and HPSG

The relation of constructional inheritance has an empirical interpretation analogous to that of subtype in HPSG although the formal notion of reentrancy, which is central to HPSG, plays no role in this CG architecture. Often a CG analysis can be directly translated into an HPSG analysis (and vice versa) by positing an HPSG type for each CG construction and assuming isomorphism between the relevant part of the type hierarchy in HPSG and a combination of the relevant part of the constructional inheritance hierarchy together with certain on-line combinations in CG.

The formal conception of CG architecture sketched in these notes is, however, more similar to that of LFG. A separation is maintained between constituency and other syntactic properties in the modeling space. The constraint equations which populate the description space in CG are quite similar to those of LFG. One difference between LFG and CG is the way the full set of constraint equations characterizing a model object is factored by the grammar. In LFG the factoring is into two subsets, one a set of (annotated) PS equations (= rules), the other a set of lexical items (containing semantic forms). Various principles ('up and down arrow' computations) provide for combining these into the full set of equations licensing the construct. CG, on the other hand, designates as linguistic units small subsets of constraint equations (the constructions); these contain both C-structure and non-C-structure information. CG resembles HPSG in this regard, in endowing the constructions (for HPSG: types) with semantic, including 'pragmatic', and phonological information, in addition to the syntactic information.

A formal difference between CG and (I think) both LFG and HPSG is that CG admits only actual words, phrases and sentences (constructs) as elements of the modeling domain. In

HPSG terms, this would roughly amount to saying that the only modeling domain objects are signs. Abstractions over actual linguistic objects (words, phrases, sentences, i.e., constructs) are not considered to be part of the modeling domain in CG.

## 8.1 Architecture and Theory

What has been sketched here is a formal architecture in which a theory of construction grammar can be expressed, not a theory of construction grammar. Within this architecture any sort of construction can be expressed; for example, we could express a construction for a transitive sentence whose subject begins with a vowel and whose object contains a relative clause specifying the color of its denotatum. A properly constrained constructional theory of grammar will have something to say about the likelihood of such a construction being encountered in any language. But that is a matter of theory, not of underlying architecture. To be sure, an architecture imposes constraints on the theories that can be expressed within it, but we should not expect a theory of grammar to follow from the notation in which it is expressed any more than we expect a theory of planetary motions to follow from the notation of differential equations.

# References


FILLMORE, CHARLES J. 1998. Inversion and constructional inheritance. In *Lexical and Constructional Aspects of Linguistic Explanation*, ed. by Gert Webelhuth, Jean-Pierre Koenig, & Andreas Kathol, volume 1 of *Studies in Constraint-Based Lexicalism*, 113–128. Stanford, CA: CSLI Publications.

——, & PAUL KAY. Forthcoming. *Construction Grammar*. Stanford: CSLI Publications.

KAPLAN, RONALD. 1995. The Formal Architecture of Lexical Functional Grammar. In *Formal Issues in Lexical-Functional Grammar*, ed. by Mary Dalrymple, Ronald Kaplan, John Maxwell III, & Annie Zaenen, 7–27. Stanford, CA: CSLI Publications.

——, & JOAN BRESNAN. 1982. Lexical Functional Grammar: A formal system for grammatical representation. In *The Mental Representation of Grammatical Relations*, 173–281. Cambridge, MA: MIT Press. Also appeared in Dalrymple et al., 1995, *Formal Issues in Lexical-Functional Grammar*, CSLI Publications, Stanford, 29–130.

KAY, PAUL, & CHARLES J. FILLMORE. In Press. Grammatical Constructions and Linguistic Generalizations: the *What's X doing Y?* Construction. *Language* .

POLLARD, CARL J., & IVAN A. SAG. 1987. *Information-based Syntax and Semantics*. Volume 1, CSLI Lecture Notes Series No. 13. Stanford, CA: CSLI Publications.

——, & ——. 1994. *Head-Driven Phrase Structure Grammar*. Chicago, IL: University of Chicago Press and Stanford: CSLI Publications.

SAG, IVAN A. 1997. English Relative Clause Constructions. *Journal of Linguistics* 33.431–484.