How is language structure built, and why?

Edward Stabler and Chris Collins

ESSLLI09
University of Bordeaux, July 24, 2009

- descriptive adequacy
- explanatory adequacy

What is Merge? 'MG-derivationalism' says...

- **vocabulary** $\Sigma = \{$every,some,student,...$\}$
- **syntactic features** F:

c, t, d, n, v, p,...	(selected categories)
=c, =t, =d, =n, =v, =p,...	(selector features)
+wh, +case, +focus,...	(licensors)
-wh, -case, -focus,...	(licensees)

- **lexicon** $Lex \subset F^*\Sigma^*$, a finite set

(Lex is the only part of the grammar that varies across languages)

Notation:

$t[f]$ = tree with 1st feature f at its head,
t = result of removing f

$t\{t_1/t_2\}$ = the result of replacing t_1 by t_2 in t

$t_1^>$ = the maximal projection of the head of t_1

ϵ = the 1 node tree labeled with no syntactic or phonetic features (sometimes we nodes with ϵ unlabeled altogether)

- **expressions** E: trees with non-root nodes $\{<,>\}$, leaves $F^*\Sigma^*$
- $Lex \subset F^*\Sigma^*$, a finite set of 1-node trees
- **Merge = {em,im}**:

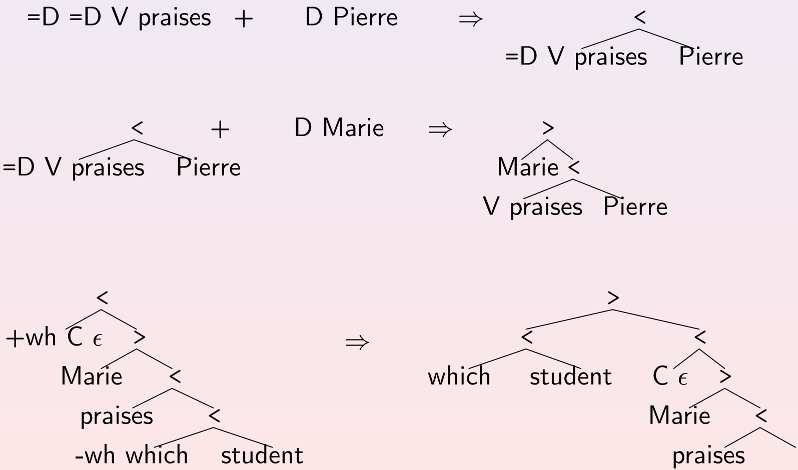
$$\mathbf{em}(t_1[\texttt{=c}], t_2[\texttt{c}]) = \begin{cases} [_{<}\ t_1\ t_2] & \text{if } t_1 \text{ has exactly 1 node} \\ [_{>}\ t_2\ t_1] & \text{otherwise} \end{cases}$$

$$\mathbf{im}(t_1[\texttt{+f}]) = [_{>}\ t_2^{>}\ \ t_1\{t_2[\texttt{-f}]^{>}/\epsilon\}]$$

if (SMC) only <u>one</u> head has -f as its first feature

Other special operations can be added (head-movement, merge-left, merge-right, scramble, adjoin, late-adjoin,. . .)

What is Merge? ‘MG-derivationalism’ says. . .



What is Merge? are there simpler answers?

- (Min1)

  Merge(A,B)={A,B}; IM(A,B)={A,B} when B contained in A

- (Min2)

  Merge(A,B)={A,B}; IM(A,B)={A,B} when B a copy of something contained in A, sometimes with modifications

Let's try developing (Min1), and compare it to MG.

Def. **Universal Grammar** is a 6-tuple:
$\langle \text{PHON}, \text{SYN}, \text{SEM}, \text{Select}, \text{Merge}, \text{Transfer} \rangle$
PHON, SYN and SEM are universal sets of features.
Select, Merge and Transfer are universal operations.

Def. A **lexical item** is a triple of three sets of features,
$\text{LI} = \langle \text{Sem}, \text{Syn}, \text{Phon} \rangle$ where Sem $\subseteq$ SEM, Syn $\subseteq$ SYN,
and Phon $\subseteq$ PHON.

Def. A **lexicon** is a set of lexical items.

Def. An **I-Language** is a pair $\langle \text{LEX}, \text{UG} \rangle$ where LEX is a lexicon
and UG is Universal Grammar.

(Chomsky'95: 227) "But the syntactic objects formed by distinct applications of Select to LI must be distinguished; two occurrences of the pronoun *he*, for example, may have entirely different properties at LF. l and l' are thus marked as distinct for CHL if they are formed by distinct applications of Select accessing the same lexical item of N."

Def. A lexical item token is a pair $\langle \mathrm{LI}, k \rangle$ where LI is a lexical item and k is an integer.
We often write $John_k$ for $\langle John, k \rangle$.

Def. A **lexical array** (LA) is a finite set of lexical item tokens.

Def. X is a **syntactic object** iff

a. X is a lexical item token, or

b. X is a set of syntactic objects.

Def. A **stage** (of a derivation) is a pair $S = \langle \text{LA}, \text{W} \rangle$, where LA is a lexical array and W is a set of syntactic objects, the "workspace" of S.

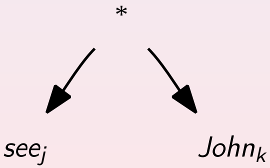
Def. Consider any stage $S = \langle LA, W \rangle$ with $LI \in LA$, then

$$\textbf{Select}(LI, S) = \langle LA - \{LI\}, W \cup \{LI\} \rangle.$$

Def. Let W be a workspace with $A, B \in W, A \neq B$. Then, **External-Merge**$_W(A, B)$ is defined as follows:

$$\text{EM}_W(A, B) = \{A, B\}.$$

Ex. $\text{EM}_W(see_j, John_k) = \{see_j, John_k\}$. Writing $*$ for the set $\{see_j, John_k\}$, the membership relation gives us this binary branching tree:



Def. For stages $S1 = \langle \mathrm{LA}_1, \mathrm{W}_1 \rangle$ and $S2 = \langle \mathrm{LA}_1, \mathrm{W}_2 \rangle$ with distinct $A, B \in \mathrm{W}_1$, $S1$ **derives** $S2$ **by External-Merge** iff

$$\mathrm{W}_2 = ((\mathrm{W}_1 - \{A, B\}) \cup \{\mathrm{EM}_{W_1}(A, B)\}).$$

Ex. $\langle \mathrm{LA}_1, \{A, B\} \rangle$ derives $\langle \mathrm{LA}_1, \{\{A, B\}\} \rangle$ by EM.
$\langle \mathrm{LA}_1, \{A, B, C\} \rangle$ derives $\langle \mathrm{LA}_1, \{\{A, B\}, C\} \rangle$ by EM.

Def. B **immediately contains** A iff $A \in B$.
B **contains** A iff

a. B immediately contains A, or

b. For some syntactic object C, B immediately contains C and C contains A.

So "immediately contains" is the "has as a member" relation, and "contains" is the transitive closure of that relation.

Def. A and B are **sisters** in C iff $A, B \in C$.

Def. A **c-commands** B iff for some C

i. C is a sister of A, and

ii. either $B = C$ or C contains B.

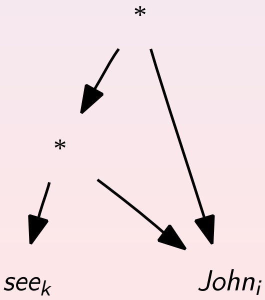
Def. For workspace W where $A \in W$ and A contains B, define **Internal-Merge**$_W(A, B)$

$$\mathrm{IM}_W(A, B) = \{A, B\}.$$

Ex. With $W = \{\{John_i, see_k\}\}$,

$$\text{IM}_W(\{John_i, see_k\}, John_i) = \{\{John_i, see_k\}, John_i\}.$$

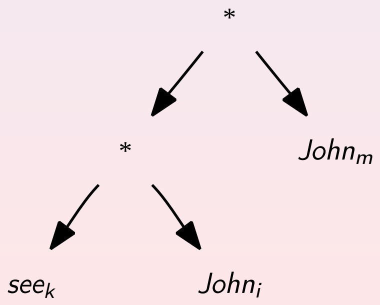
Note that the membership diagram for $\{\{John_i, see_k\}, John_i\}$ is not a tree:



Ex. With $W = \{\{John_i, see_k\}, John_m\}$,

$$\text{EM}_W(\{John_i, see_k\}, John_m) = \{\{John_i, see_k\}, John_m\}.$$

The membership diagram for $\{\{John_i, see_k\}, John_m\}$ is a tree:



Def. For stages $S1 = \langle \mathrm{LA}_1, \mathrm{W}_1 \rangle$ and $S2 = \langle \mathrm{LA}_1, \mathrm{W}_2 \rangle$ where $A \in \mathrm{W}_1$ and A contains B, $S1$ **derives** $S2$ **by Internal-Merge** iff

$$\mathrm{W}_2 = ((\mathrm{W}_1 - \{A\}) \cup \{\mathrm{IM}_{W1}(A, B)\}).$$

Here B "undergoes internal merge, targeting A", to form $\{A, B\}$.

- When EM derives W_2 from W_1, $|W_2| = |W_1| - 1$.
  When IM derives W_2 from W_1, $|W_2| = |W_1|$.

(Chomsky'00: 115) proposes two ways to define the position of an occurrence in a structure.

First, he takes "...an occurrence of α in K to be the full context of α in K."

Alternatively, he suggests "...an occurrence of α is a sister of α."

Def. A **path** is a sequence of syntactic objects $\langle \mathrm{SO}_1, \mathrm{SO}_2, \ldots, \mathrm{SO}_n \rangle$, where for all $0 < i \leq n$, $\mathrm{SO}_{i+1} \in \mathrm{SO}_i$

The **position** of SO_n in SO_1 is a path $\langle \mathrm{SO}_1, \mathrm{SO}_2, \ldots, \mathrm{SO}_n \rangle$.

B **occurs** in A at position P iff $P = \langle A, \ldots, B \rangle$. We also say B has an occurrence in A at position P (written B_P).

Ex. Consider $\mathrm{SO} = \{X, \{X, Y\}\}$, where X occurs twice:

The higher occurrence of X in SO is X_P where

$$P = \langle\ \mathrm{SO},\ X\ \rangle.$$

The lower occurrence of X in SO is X_P where

$$P = \langle\ \mathrm{SO},\ \{X, Y\},\ X\ \rangle.$$

Def. Let A, B and C be syntactic objects, then, in C, **occurrence B_P immediately contains occurrence $A_{P'}$** (for any paths P, P' in C) iff $P = \langle X_1, \ldots X_n \rangle$ and $P' = \langle X_1, \ldots, X_n, X_{n+1} \rangle$.

Thm. If occurrence B_P immediately contains occurrence $A_{P'}$ in C (for some paths P, P' in C) then, in C, B immediately contains A.

- We can similarly define sister, c-command for occurrences.

Def. For stages, $S1 = \langle \mathrm{LA}_1, \mathrm{W}_1 \rangle$ and $S2 = \langle \mathrm{LA}_2, \mathrm{W}_2 \rangle$, $S1$ **derives** $S2$ iff

i. $S1$ derives $S2$ by EM, or

ii. $S1$ derives $S2$ by IM, or

iii. for some $\mathrm{LI} \in \mathrm{LA}_1$, $S2 = \mathrm{Select}(\mathrm{LI}, S1)$.

Def. Sequence of stages $\langle \langle \mathrm{LA}_1, \mathrm{W}_1 \rangle, \ldots, \langle \mathrm{LA}_n, \mathrm{W}_n \rangle \rangle$, is **derivation** from lexicon L iff

a. For all $\langle \mathrm{LI}, k \rangle \in \mathrm{LA}_1$, $\mathrm{LI} \in L$,

b. $W_1 = \emptyset$

c. for all $1 \leq i \leq n - 1$, $\langle \mathrm{LA}_i, \mathrm{W}_i \rangle$ derives $\langle \mathrm{LA}_{i+1}, \mathrm{W}_{i+1} \rangle$,

d. $\mathrm{LA}_n = \emptyset$, and

e. W_n contains exactly one element.

Ex. Derivation of "John should like John."

$S_1 = \langle\{\text{John}_1, \text{should}_2, \text{like}_3, \text{John}_4\}, \emptyset\rangle$	Select John_4
$S_2 = \langle\{\text{John}_1, \text{should}_2, \text{like}_3\}, \{\text{John}_4\}\rangle$	Select like_3
$S_3 = \langle\{\text{John}_1, \text{should}_2\}, \{\text{like}_3, \text{John}_4\}\rangle$	Merge(like_3, John_4)
$S_4 = \langle\{\text{John}_1, \text{should}_2\}, \{\{\text{like}_3, \text{John}_4\}\}\rangle$	Select should_2
$S_5 = \langle\{\text{John}_1\}, \{\text{should}_2, \{\text{like}_3, \text{John}_4\}\}\rangle$	Merge(should_2, $\{\text{like}_3$, Jo
$S_6 = \langle\{\text{John}_1\}, \{\{\text{should}_2, \{\text{like}_3, \text{John}_4\}\}\}\rangle$	Select John_1
$S_7 = \langle\emptyset, \{\text{John}_1, \{\text{should}_2, \{\text{like}_3, \text{John}_4\}\}\}\rangle$	Merge($\text{John}_1, \ldots.$)
$S_8 = \langle\emptyset, \{\{\text{John}_1, \{\text{should}_2, \{\text{like}_3, \text{John}_4\}\}\}\}\rangle$	

Def. Syntactic object A is **binary branching** iff A and every B contained in A is either a lexical item or a syntactic object immediately containing exactly two syntactic objects.

Thm. (Binary branching) Every derivable syntactic object is binary branching.

Thm. (Uniqueness of root occurrences) In every derivable workspace W, if $A \in$ W there is no $B \in$ W such that A contains an occurrence in B.

Example underivable workspace : $\{A, \{A, B\}\}$

Thm. (Non-unique occurrences signal IM) A derivable workspace contains two distinct occurrences of A iff either A or some B containing A has undergone IM.

(Chomsky'05: 5) "Suppose X and Y are merged. Evidently, efficient computation will leave X and Y unchanged (the No-Tampering Condition NTC). We therefore assume that NTC holds unless empirical evidence requires a departure from SMT in this regard, hence increasing the complexity of UG. Accordingly, we can take Merge(X, Y) = {X, Y}."

(Chomsky'05: 13) "The no-tampering condition also entails the so-called copy theory of movement, which leaves unmodified the objects to which it applies, forming an extended object."

Thm. (No Tampering Condition) For any two consecutive stages in a derivation $S_1 = \langle \mathrm{LA}_1, \mathrm{W}_1 \rangle$ and $S_2 = \langle \mathrm{LA}_2, \mathrm{W}_2 \rangle$, for all $A \in \mathrm{W}_1$, either $A \in \mathrm{W}_2$, or there is some $C \in \mathrm{W}_2$ and $A \in C$.

Thm. (Extension Condition) For any two consecutive stages $S_1 = \langle \mathrm{LA}_1, \mathrm{W}_1 \rangle$ and $S_2 = \langle \mathrm{LA}_2, \mathrm{W}_2 \rangle$, if S_1 derives S_2 by Merge (External-Merge or Internal-Merge), then there is some $A \in \mathrm{W}_1$ and $C \in \mathrm{W}_2$ such that

i. $C \notin \mathrm{W}_1$ (C is created by Merge)
ii. $A \notin \mathrm{W}_2$ (A is extended)
iii. $A \in C$. (A is extended to form C)

(Chomsky'95: 225) "Another natural condition is that outputs consist of nothing beyond properties of items of the lexicon (lexical features)...other words, that the interface levels consist of nothing more than arrangements of lexical features."

Thm. (Inclusiveness) In any derivation

$$\langle\langle \mathrm{LA}_1, \mathrm{W}_1\rangle, \ldots, \langle \mathrm{LA}_n, \mathrm{W}_n\rangle\rangle$$

where $\mathrm{W}_n = \{A\}$, the only elements contained in W_n are the lexical item tokens from LA_1 and sets containing them.

(Chomsky'95: 227) : "I and I' are marked as distinct for CHL if they are formed by distinct applications of Select accessing the same lexical item of N. Note that this is a departure from the inclusiveness condition, but one that seems indispensable; it is rooted in the nature of language, and perhaps reducible to bare output conditions."

Thm. (Collins 1997: 4) Whether or not Merge applies is determined completely by the workspace and the syntactic objects it contains.

Collins calls this **local economy**: whether or not Merge applies never depends on information contained in another workspace (from a stage either earlier or later in the derivation, or from a different derivation altogether).

(Chomsky'08: 139) "For an LI to be able to enter into a computation, merging with some SO, it must have some property permitting this operation. A property of an LI is called a feature, so an LI has a feature that permits it to be Merged. Can this the edge feature (EF) of the LI."

Def. A lexical item token $LI = \langle\langle \text{Sem}, \text{Syn}, \text{Phon}\rangle, i\rangle$ **contains** an edge feature EF iff some $EF \in Syn$ is an edge feature.

(Chomsky'00: 132) "Properties of the probe/selector a must be satisfied before new elements of the lexical subarray are accessed to drive further operations."

Def. If LI has an EF token, $\text{Select}(\text{LI}, \langle \text{LA}, \text{W}\rangle)$ is defined only if there are no other syntactic objects $A \in \text{W}$ containing an EF token.

Def. Triggers maps each derivable syntactic object A to its 'visible' edge features:

a. If A is a lexical item, then Triggers(A) returns all its edge features.

b. If $A = \{B, C\}$, Triggers(B) is nonempty, and Triggers(C) $= \emptyset$, then Triggers(A) $=$ Triggers(B) $- \{F\}$, for some edge feature token F.

Def. Let W be a workspace and $A, B \in W$, Triggers(A) contains $n > 0$ edge features and Triggers(B) is empty. Then, (triggered) $\text{EM}_W(A, B) = \{A, B\}$.

Def. Let W be a workspace where $A \in W$ and A contains B, where Triggers(A) is nonempty and Triggers(B) is empty. Then, (triggered) $\text{IM}_W(A, B) = \{A, B\}$.

Example. Suppose see_1 has 2 edge features, and $John_2$ has 0 edge features.

$S_1 = \langle\{\text{John}_1, \text{see}_2\}, \emptyset\rangle$	Select John_1
$S_2 = \langle\{\text{see}_2\}, \{\text{John}_1\}\rangle$	Select see_2
$S_3 = \langle\emptyset, \{\text{John}_1, \text{see}_2\}\rangle$	$\text{Merge}(\text{see}_2, \text{John}_1)$
$S_4 = \langle\emptyset, \{\{\text{John}_1, \text{see}_2\}\}\rangle$	$\text{Merge}(\text{John}_1, \{\text{John}_1, \text{see}_2\})$
$S_5 = \langle\emptyset, \{\{\text{John}_1, \{\text{John}_1, \text{see}_2\}\}\}\rangle$	

Thm. Considering triggered EM, if $\text{EM}(A, B)$ is defined, $\text{EM}(B, A)$ is undefined. Similarly for triggered IM.

Def. (cf.Collins'02) For all lexical item tokens LI, Label(LI) = LI. For any other SO $\{A, B\}$, if Triggers(A) is non-empty, then Label($\{A, B\}$) = Label(A).

Def. C is a **maximal projection** of LI iff Label(C) = LI and there is no D in W which immediately contains C such that Label(D) = Label(C).

Def. For all C, C is a **minimal projection** iff C is a lexical item token.

Def. C is an **intermediate projection** of LI iff Label(C) = LI, and C is neither a minimal projection nor a maximal projection in W.

Def. Y is the **complement** of X in C iff C = Merge(X, Y) and X is a lexical item token.

Def. Y is the **specifier** of X in C iff C = Merge(X, Z) where X is not a lexical item token.

Def. For any derivable workspace W with syntactic object $\text{Phase} \in W$ such that $\text{Label(Phase)} = X$ is a strong phase head, let Y be the complement of X in Phase, then $\text{Cyclic-Transfer(Phase)} = \text{Phase}'$ where Phase' is obtained from Phase by replacing $\{X, Y\}$ in Phase by $\{X, \langle \text{Transfer}_{PF}(Y), \text{Transfer}_{LF}(Y)\rangle\}$.

Def. X is a **strong phase head** iff X is C or v.

Def. X is a **syntactic object** iff

a. X is a lexical item token,

b. $X = \text{Cyclic-Transfer}(SO)$ for some syntactic object SO, or

c. X is a set of syntactic objects.

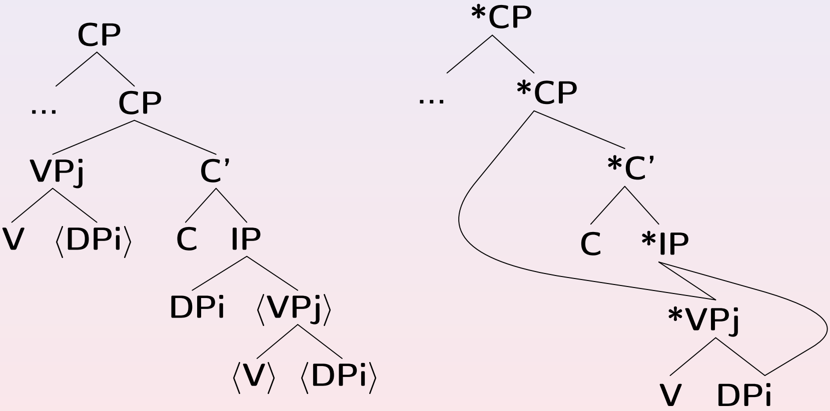
(Chomsky'05: 13) "If language is optimized for satisfaction of interface conditions, with minimal computation, then only one [copy] will be spelled out, sharply reducing phonological computation."

We adopt this idea and describe a simple approach for illustration. . .

Def. X_P is **nonfinal** in Y_Q iff Y_Q contains Z_R and Z_S such that

a. (i) either $Z_R = X_P$ or Z_R contains X_P, and (ii) Z_S c-commands Z_R, or

b. (i) Z_R contains X_P, (ii) Z_R c-commands Z_S, which contains another occurrence $X_{P'}$, and (iii) $X_{P'}$ is nonfinal in the sister of Z_R.



E.g. node labeled V on left is final because (i) neither V nor any object containing V is c-commanded by any occurrence of itself, and (ii), because although (a) VPj does c-command ⟨VPj⟩, V is not nonfinal in C'.

Def. For derivable workspace W with syntactic object Phase where Label(Phase) is a strong phase head, and for all occurrences of objects SO such that either SO = Phase or SO is contained in Phase,

a. If SO is LI, then $\text{Transfer}_{PF}(SO) = \text{Phon}$;

b. If $SO = \{X, Y\}$, occs X and Y both final in Phase, $\text{Transfer}_{PF}(SO) = \text{Transfer}_{PF}(X)^\frown \text{Transfer}_{PF}(Y)$ if either Y is the comp of X, or X is the spec of Y;

c. If $SO = \{X, Y\}$, occ X but not Y final in Phase, $\text{Transfer}_{PF}(SO) = \text{Transfer}_{PF}(X)$ if

d. If $SO = \{X, Y\}$, occ Y but not X final in Phase, $\text{Transfer}_{PF}(SO) = \text{Transfer}_{PF}(Y)$ if

e. If $SO = \{X, Y\}$, neither X nor Y final in Phase, $\text{Transfer}_{PF}(SO) = \epsilon$

Problem:

- Transfer$_{PF}$ requires seeing into transferred phases

To avoid this problem, there are other options besides Min2 and MG-style-derivationalism.

Not treated here:

- Agree
- Transfer$_{PF}$ for multiple spellout
  (cf., e.g., Hiraiwa'05, Kobele'06, Kandybowicz'08)
- Transfer$_{LF}$ for reconstruction effects

Min:

- IM(A,B)={A,B} with B contained in A
- No need for 'occurrences' etc in def of EM, IM
- Trees represented as nested sets; IM produces multidominance
- Edge feature calculation for A traces path to leaf
- Transfer_{PF} reconstructs derivation, violates NTC, vs Phases

MG:

- IM(A,B)=[B,A'], where A' has B deleted (violates NTC)
- No need for 'occurrences' etc in def of EM, IM
- Trees OK; no multidominance needed (since B deleted in A')
- Edge features explicit, ordered; Transfer_{PF} trivial; Phases OK

What is Merge?

- Merge = {em,im}

$$\textbf{em}(t_1[\text{=c}], t_2[\text{c}]) = \begin{cases} \genfrac{}{}{0pt}{}{<}{t_1 \quad t_2} & \text{if } t_1 \text{ has exactly 1 node} \\ \genfrac{}{}{0pt}{}{>}{t_2 \quad t_1} & \text{otherwise} \end{cases}$$

$$\textbf{im}(t_1[\text{+f}]) = \genfrac{}{}{0pt}{}{>}{t_2^{>} \quad t_1\{t_2[\text{-f}]^{>}/\epsilon\}} \qquad \text{if (SMC) only \underline{one} head has -f as its first feature}$$

- Merge(A,B)={A,B}; IM(A,B)={A,B} when B contained in A

(cf alternatives: TAG, CG,. . .)

Chomsky, Noam. 1995. The Minimalist Program. MIT Press, Cambridge, Massachusetts.

Chomsky, Noam. 2000. Minimalist inquiries: The framework. In Roger Martin, David Michaels, and Juan Uriagereka, editors, Step by Step: Essays on Minimalism in Honor of Howard Lasnik. MIT Press, Cambridge, Massachusetts, pages 89–155.

Chomsky, Noam. 2005. Three factors in language design. Linguistic Inquiry, 36(1).

Chomsky, Noam. 2008. On phases. In Robert Freidin, Carlos P. Otero, and Maria Luisa Zubizarreta, editors, Foundational Issues in Linguistic Theory: Essays in Honor of Jean-Roger Vergnaud. MIT Press, Cambridge, Massachusetts.

Collins, Chris. 1997. Local Economy. MIT Press, Cambridge, Massachusetts.

Collins, Chris. 2002. Eliminating labels. In Samuel D. Epstein and Daniel Seeley, editors, Derivation and Explanation. Blackwell, Oxford.

Hiraiwa, Ken. 2005. Dimensions of Symmetry in Syntax: Agreement and Clausal Architecture. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Kandybowicz, Jason. 2008. The Grammar of Repetition: Nupe grammar at the syntax-phonology interface. John Benjamins, Philadelphia.

Kobele, Gregory M. 2006. Generating Copies: An Investigation into Structural Identity in Language and Grammar. Ph.D. thesis, UCLA.

Stabler, Edward P. 1997. Derivational minimalism. In Christian Retoré, editor, Logical Aspects of Computational Linguistics. Springer-Verlag (Lecture Notes in Computer Science 1328), NY, pages 68–95.