

# Minimalist Syntax of German(ic)

University of Illinois at Chicago, volker.struckmeier@uni-koeln.de

## Part I: Minimalist vs. GB derivations

### I.1. General architecture of minimalist syntax

The *Minimalist Program (MP)* is concerned with an *evolutionary* explanation of syntax. This can only be achieved if syntax can (at least potentially) result from a minor, random mutation that differentiates homo sapiens sapiens from all other hominids. The overall aim, then, is to simplify syntax as much as possible. In the process, many operations formerly handled by syntax are now delegated to other linguistic subsystems, i.e. the lexicon, PF/LF, or are considered extralinguistic altogether.

- The *lexicon* of a language is composed of *lexical items (LI)*.
- An LI is basically a *set of features* which, taken together, make up the word, morpheme, etc: {phonological features, semantic features, morphosyntactic features}
- According to the conception of the lexicon advocated here, LIs also comprise *all inflected forms*.
- Hence, syntax does not have to assemble words, e.g., finite verbs from stem + affix. I.e., the derivation only *licenses* a pre-assembled LI's features.
- Syntax only consists of two operations, *Merge* and *Agree*.
- Merge and Agree apply *freely any number of times*:

*Merge* applies until all LIs used in a given derivation are assembled into one syntactic object (i.e. the sentence)

*Agree* applies until all features included in the LIs have been licensed (i.e. all agreement properties have been taken care of).

- The derived single object has to be read by the *interface systems*, LF and PF.
- LF and PF take over some tasks that were handled syntactically in GB derivations.

Unlike GB, MP seeks to *avoid the ordering of operations*. This means that all levels of representation other than LF and PF (the bare minimum, if a sentence has a pronunciation and a meaning) are abolished:

- Surface structure as the *only* point in a derivation for filters etc. to apply is eliminated.
- Deep structure as the *only* point for X-bar, theta theory etc. is, too.

So, how can the system work without all the tools that made GB tick?

### I.2. Building a tree bottom-up: Merge

Rather than eliminating syntactic operations one by one, let's start the other way round: What is the minimal, least complex operation that we need in order to do syntax at all?

Sentences are composed of smaller elements. Hence, an operation that assembles sentences from smaller components is conceptually necessary. Call this operation *Merge* (Chomsky 2005: 4, 2006: 4):

- *Merge* takes a syntactic object A, and a syntactic object B as its input.
- LIs are primitive syntactic objects.
- *Merge* combines two objects into a new object, i.e. a set that contains A and B as elements:
  - a)  $Merge(A, B) \rightarrow \{A, B\}$
- Given the new object, *Merge* can recursively add objects:
  - b)  $\{A, B\}, C \rightarrow \{\{A, B\}, C\}$
  - c)  $\{\{A, B\}, C\}, D \rightarrow \{\{\{A, B\}, C\}, D\}$
  - ...
- *Merge* that takes LIs from the lexicon as its input is also called *External Merge*: 'new' objects are *added* to the syntactic object under construction, i.e. objects which were *not* included in the object prior to *Merge*.

In the simplest case, *Merge* combines two objects -- and does absolutely *nothing else*. *Merge* then has the following desirable properties:

- *Inclusiveness* (cf. Chomsky 2000: 112, 2006: 4): *Merge cannot introduce features* into the derivation that do not come as a part of an LI.

This implies that bar levels, indices etc. cannot be used in a derivation anymore (resulting in

a *bare phrase structure*).

This, in turn, implies that the 'projection level' ( $X^0$ ,  $X'$ ,  $X''$ ) of an object can only be *computed from the object's context*:

If the object is not itself constructed by *Merge*, it is a *head*.

If an object does not project further, it is a *phrase*.

(Projection is defined in terms of selectional features of an LI)

- *No tampering* (cf. Chomsky 2005: 5): *Merge cannot alter features* that have been introduced as part of a syntactic object.

This also implies that *Merge* can only apply at the 'root' of a syntactic tree: the object already constructed cannot be altered any more (cf. Chomsky 2006: 8).

- *Merge does not determine linear ordering* of the elements merged:

"order does not enter into the generation of [LF], and [...] syntactic determinants of order fall within the phonological component" (Chomsky 2005: 5).

### Derivation of "Peter sieht die Katze"

Assume that the elements to be merged together are:  $\{v, T, C, \text{Peter}, \text{sieht}, \text{die}, \text{Katze}\}$ . The derivation then starts from this *numeration* and proceeds as follows:

*Die* has a selectional feature for an N complement. Hence:

(i)  $\text{die}, \text{Katze} \rightarrow \{\text{die}, \text{Katze}\}$

*sieht*, in turn, has a requirement for a theme argument:

(ii)  $\{\text{die}, \text{Katze}\}, \text{sieht} \rightarrow \{\{\text{die}, \text{Katze}\}, \text{sieht}\}$

In order to license the accusative of *die Katze*, the functional head *v* has to be merged:

(iii)  $v, \{\{\text{die}, \text{Katze}\}, \text{sieht}\} \rightarrow \{v, \{\{\text{die}, \text{Katze}\}, \text{sieht}\}\}$

$v$  furthermore requires a subject to be positioned in its specifier:

(iv)  $Peter, \{v, \{\{die, Katze\}, sieht\}\} \rightarrow \{Peter, \{v, \{\{die, Katze\}, sieht\}\}\}$

T selects for a  $vP$  complement:

(v)  $T, \{Peter, \{v, \{\{die, Katze\}, sieht\}\}\} \rightarrow \{\{Peter, \{v, \{\{die, Katze\}, sieht\}\}\}, T\}$

At this point, only C is available for *External Merge*. However, T requires a specifier, too (EPP!). Given that C is a head (it has not been assembled by *Merge*), it is not a suitable object to be merged at this point. So how do we move a phrase into SpecTP?

### 1.3. Movement is *Merge*, too

In the MP, movement is implemented via the operation *Merge*, too: Given that *Merge* can merge *any* syntactic object to the root of a syntactic tree, it can also apply to an object that already *is* a part of the constructed tree. Call this *Internal Merge*.

However, given *No tampering* and *Inclusiveness*, the object already built must not be changed by this process: No *traces* etc. can be used to signal the starting position of the movement process. So how can the movement be represented?

Note that *Merge* does not 'take' LIs 'out of' the lexicon: After an LI is merged, it is of course still present in the lexicon. Hence, *Merge* always comprises a subcomponent called *Copy* (Chomsky 2005: 6). An LI is copied from the numeration in order to be used in a derivation. If movement is indeed *Merge*, *Copy* should be part of *Internal Merge*, too:

(vi)  $\{\{A, B\}, C\} \rightarrow \{A, \{\{A, B\}, C\}\}$

The different tasks of X-bar and Move are delegated to LF (cf. Chomsky 2005: 7):

- LF *interprets* External Merge as relating to theta configurations.
- LF *interprets* Internal Merge as indicating information structure, scopal properties etc..

The result of the example derivation up to (v) was:

(vi)  $\{\{\text{Peter}, \{v, \{\{\text{die}, \text{Katze}\}, \text{sieht}\}\}\}, T\}$

Internal Merge can now move *sieht* to its (intermediate) position in T by copying it and placing the copy in T:

(vii)  $\{\{\text{Peter}, \{v, \{\{\text{die}, \text{Katze}\}, \text{sieht}\}\}\}, \text{sieht-T}\}$

The specifier of T can only be merged internally, too:

(ix)  $\{\{\text{Peter}, \{v, \{\{\text{die}, \text{Katze}\}, \text{sieht}\}\}\}, \text{sieht-T}\} \rightarrow$   
 $\{\text{Peter}, \{\{\text{Peter}, \{v, \{\{\text{die}, \text{Katze}\}, \text{sieht}\}\}\}, \text{sieht-T}\}\}$

T has to project a specifier in every derivation: T's feature set comprises a so-called *EPP* feature, which essentially states that a phrase should project a specifier (see 1.4 below).

In the remainder of the derivation, C is externally merged:

(x)  $C, \{\text{Peter}, \{\{\text{Peter}, \{v, \{\{\text{die}, \text{Katze}\}, \text{sieht}\}\}\}, \text{sieht-T}\}\} \rightarrow$   
 $\{C, \{\text{Peter}, \{\{\text{Peter}, \{v, \{\{\text{die}, \text{Katze}\}, \text{sieht}\}\}\}, \text{sieht-T}\}\}\}$

*sieht* then internally merges a second time, resulting in:

(xi)  $\{\textit{sieht-C}, \{\textit{Peter}, \{\{\textit{Peter}, \{v, \{\{\textit{die}, \textit{Katze}\}, \textit{sieht}\}\}\}, \textit{sieht-T}\}\}\}$

Likewise, the object *Peter* can internally merge to its final position in SpecCP:

(xii)  $\{\textit{Peter}, \{\textit{sieht-C}, \{\textit{Peter}, \{\{\textit{Peter}, \{v, \{\{\textit{die}, \textit{Katze}\}, \textit{sieht}\}\}\}, \textit{sieht-T}\}\}\}$

As all LIs needed are used up, the derived object is handed over to the interfaces:

- LF interprets (only) the hierarchical configuration achieved.
- PF determines, which copies are pronounced and linearly orders the syntactic tree.

So far, we haven't talked about the agreement properties of the sentence. Enter *Agree*.

#### 1.4. The operation *Agree*

Given *Inclusiveness* and *No tampering*, features cannot be assigned to objects in the way that they could in GB. Thus, all that agreement has to do is to *license* the features introduced by the LIs.<sup>1</sup> The operation that implements this is called *Agree*.

The operation *Agree* can occur between two syntactic objects, iff one object *includes* the other, as in:

(xiii) [A... A ... B ... ] *or* [B ... A ... B ... ]

---

<sup>1</sup> Note that there is different conceptions about this process, e.g. *Agree* could *copy feature values* between objects etc. We will assume here (somewhat simplifying) that *Agree* simply confirms that the relevant features on the relevant LIs do agree with one another.

*Agree cannot* occur when *no* object contains the other:

(xiv) [c [c ...A...] [D ... B...]]

The two parties in the *Agree* process can be referred to by their role:

- The set of features of the contained element (i.e. the one 'lower in the tree') is the *goal*.
- The relevant set of the containing object ('higher in the tree') is the *probe*.

Probe features are associated with specific heads (optimally, *phase heads*, see 1.5):

- C contains features which license the case of the (nominative) subject DP. (These features can be inherited by (i.e. expressed on) T, yielding the correlation between verb agreement and the availability of nominative case, cf. Chomsky 2005: 9, 2006: 13, 15).
- *v* is an expression of the transitivity of a structure: Just as C/T licenses the subject's case, *v* licenses an objective case by probing its domain for an appropriate goal (cf. Chomsky 2006: 12, 15).
- *v* also introduces the sentence's subject in its specifier (cf. Chomsky 2000: 102).
- Unaccusative/ passive structures do not comprise a *v* with object-licensing features. Hence, the (only) argument in VP will receive case after C/T is merged - i.e., nominative.

*Agree* is further constrained by the requirement *Maximize Matching Effects*: *Agree* has to affect the maximal number of features available in a given operation.

For example, a finite verb should not agree with the subject in [person], but take its [number] agreement from, say, the object. This results in two types of *Agree* processes:

- If both probe and goal have the complete feature sets usually associated with them, *both*

the probe *and* the goal are licensed (*symmetrical Agree*):

A finite C/T and a regular DP contain all features associated with the Agree process:

C/T contains [person] and [number] (so-called  $\varphi$ -features),

DP contains [number], [person] and [case]

- If either probe or goal do *not* comprise the full feature set, Agree only licenses the *incomplete* feature set, but leaves the complete feature set intact:

In some languages (e.g., French, Spanish), both an auxiliary and a participle can show agreement with the subject:

Juan está cansado.

Juan es besado.

Jean est né à Paris.

María está cansada.

María es besada.

Marie est née à Paris.

The participle is regarded as an *defective* element: it lacks [person]. Hence:

The subject probe (asymmetrically) licenses the (defective) participle agreement *without* licensing its own feature set.

The subject then goes on to license the auxiliary, which has a full set of  $\varphi$ -features.

The second, symmetrical agreement process licenses the subject's features.

- How does the EPP enter into *Agree* processes? Given that the EPP states that a phrase has to project a specifier, and given also that EPP is a feature, EPP would have to be satisfied in the same (*maximal*) *Agree* process that licenses the EPP-containing head's other features: Thus, when a head includes an EPP, movement can actually be *forced* in order to satisfy *maximize Agree*:

In our example derivation, the subject DP merged into SpecvP is the only possible goal for T's  $\varphi$ -features. Given *maximal Agree*, the same phrase that licenses the  $\varphi$ -features also has to satisfy the EPP, if possible. In effect, then, merging the subject into SpecTP is forced.

## I.5. Phases and the PIC

The objects that are handed over to the interfaces for interpretation are called *phases* (cf., e.g., Chomsky 2006: 11). Phases include at least *vP* and *CP*, but probably also *DP*.

Each syntactic derivation is thus split up into smaller, partial derivations which are essentially oblivious of the ones completed: Material from one phase is not accessible to operations in later phases if it has been *spelled out*, i.e. shipped off to the interface. This results in the so-called *Phase Impenetrability Condition (PIC)*, cf. Chomsky 2000: 108, 2006: 11f.).

To give an example, consider our derivation above: The complete syntactic tree comprises both a *vP* and a *CP*, hence is made up of (at least) two phases:

- Internal *Merge* builds up *vP* as described above.
- Once *C/T* are merged, the subject of *vP* and the verb move out of *vP*.
- Thus, *VP* now only contains material that has already been taken care of in the derivation: the object's case has been licensed by *v*, and the verb has vacated *VP*.
- Thus, the complement of the phase head *v* can be shipped off for interpretation.
- To generalize this: *Spell-out* occurs to the *complement* of a phase head, once the *next* phase head is merged.<sup>2</sup>
- The specifier of a phase head is left in the derivation. This so-called *edge* of a phase is thus privileged over the phase head's complement: material in the edge can be used later on in the derivation.

How can material from the phase head's complement be saved for later use?

- Suppose the *VP* object needs to be raised, e.g. to topicalize, it has to *vacate VP before VP is shipped off*. Given that internal *Merge* is essentially 'free', this is feasible:
- The *edge* of a phase constitutes an *escape hatch* for elements to avoid spell-out.

---

<sup>2</sup> According to a different conception, spell-out might even occur earlier, i.e. the phase head ships off its complement after it has finished its own projection (Richards 2006). The difference is not important here.

## GB vs. MP at a glance: some (unrepresentative) differences

	GB:	MP:
<u>structure generation</u>	<p><u>X-bar</u></p> <p><i>projection</i> of arguments <i>projection levels</i></p> <p><u>relations</u> <i>head-complement</i> <i>spec-head</i></p> <p><u>ordering:</u> <i>head parameter</i> <i>spec parameter</i></p>	<p><u>Merge (external)</u></p> <p>LF interprets <math>\Theta</math>-structures --</p> <p><i>projecting head includes complement</i> --</p> <p>-- -- (both determined on PF only)</p>
<u>structure permutation</u>	<p><u>move alpha</u> "free"</p>	<p><u>Merge (internal)</u> ("free", too)</p>
or:	<p><u>checking</u> "greedy" move <math>\leftrightarrow</math> agreement</p> <p>only 'upward': ECP</p> <p>leaves traces</p> <p>no movement into <math>\Theta</math>-positions: <i>Theta criterion</i></p>	<p>EPP and <i>Maximize Agree</i> can force <i>internal merge to occur.</i></p> <p><i>operations apply "at the root"</i> <i>anyway = built into mechanism</i></p> <p><i>creates copies</i></p> <p><i>distinct LF-interpretations of external merge (<math>\Theta</math>-related) vs. internal merge (information structure etc.)</i></p>
Derivation is:	<p>inhomogenous: deep + surface structure "ordered" operations</p> <p><u>object generation:</u> traces bar levels</p> <p><u>object manipulation:</u> <i>feature assignment</i> <i>theta projection</i></p>	<p><i>homogenous:</i> <i>no such levels</i> <i>operations apply "any time"</i></p> <p><i>Inclusiveness = no generation copies</i> <i>'bare' phrase structure</i></p> <p><i>No tampering:</i> <i>Agree licenses LIs' features</i> <i>LF interpretation only</i></p>