Phases and Complexity in Phrase Structure Building

Cristiano Chesi

University of Siena - MIT

Abstract

The *Minimalist Program* (Chomsky 1995–2001) sketches a model that aims to be empirically adequate and theoretically motivated but that does not fit in any clear way with specific performance algorithms such as *parsing* or *generation* even though much emphasis is put on "interface properties". In this paper I propose that a *Minimalist Grammar* formalization (an extension of Stabler's 1997 proposal) can be used both in *parsing* and in *generation* if we re-orient the directionality of the *Structure Building Operations* (*merge* and *move*) and if we formalize the notion of *phase* (Chomsky 1999). This will help us to define generalized algorithms, suitable both for *parsing* and for *generation*, which are computationally tractable in dealing with *ambiguities* and *long distance dependencies* resolution.

1 Introduction to minimal grammar specification

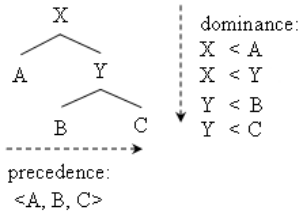
Formalizing a linguistic theory forces us to specify everything we need to describe a language. When choosing a specific formalization we must consider to which extent it encodes linguistic intuitions and at which computational cost it does. In this work I will deal essentially with this second issue,[1] providing some formal argument in favor of the necessity of limiting unbounded long distance *Structure Building Operations* (such as *move*, Chomsky 1995–2001) by means of a precise formalization of the notion of *phase* (Chomsky 1999). In order to provide the minimal background for this discussion, we need to define explicitly the notion of *Structural Description* (§1.1), a precise formalization of *Minimalist Grammar* (§1.2) and a definition of *parsing* and *generation* (§1.3). This should allow us to understand the necessity to (re)define some essential property of the *Structure Building Operations* (*merge*, *move* and the idea of *derivation by phase*, (§1.4)) because of empirical and computational considerations.

1.1 Structural Descriptions in terms of immediate relations

It is a standard assumption to consider a sentence as a bidimensional entity bearing information on both *precedence* and *dominance* relations among lexical items, where *precedence* represents a total order among pronounced elements (namely words, that are groups of phonetic features) while *dominance* expresses the constituency/dependency relations among pronounced and other implied (abstract) elements (semantic and other syntactic features like phrase identifiers). These two kinds of information can be encoded within tree-like structures such as the following one:

[1] See Chesi (2004) for a discussion of the empirical issue.

(1)



From a formal point of view, a *Structural Description* (SD) can be expressed as follows:

(2) $\mathbf{SD}_{\text{standard}}$ = {**I**, **P**, **D**, **V**, **A**} such that

I is a *precedence* order (a total strict order, that is a binary, transitive and asymmetric relation, defined on any element belonging to the subset I_T of I such that I_T is the set of terminal elements; e.g. {the, dog, is, black})

D is a set of *dominance* relations (a partial strict order, that is a binary, transitive and asymmetric relation, defined on some elements of I; e.g. "D" dominates "the", "N" dominates "dog", "DP" dominates "dog" etc.)

V is a finite set of *vertices* (the nodes in the tree)

A is an *assignment function* from V to I

I is a finite set of *identifiers* (e.g. {the, dog, is, black, DP, D′, D^o, N^o ...})

Leaving aside both **V** and **A** (maybe superfluous as discussed in Chesi 2004) I wish to redefine the other three elements in such a way that *precedence* and *dominance* only hold between immediately adjacent objects:

(3) $\mathbf{SD}_{\text{revisited}}$ = {**I**, **P**, **D**} such that

I is a finite set of *identifiers* (minimally, if we accept the inclusiveness condition, Chomsky 1995, we should consider nothing but lexical items and their features)

P is a finite set of *immediate precedence* relations (different from the classic total strict order, this relation is only defined between two adjacent items)
for example ("<A, B>" means "A *immediately precedes* B"):
SD = [$_A$ A [$_B$ B C]] → P = {<A, B>, < B, C>}

D is a finite set of *immediate dominance* relations (different from the classic *dominance* relation, this one is a partial, binary, intransitive and asymmetric relation
for example ("A < B", means "A *immediately dominates* B"):
SD = [$_A$ A [$_B$ B C]] → D = {A < B, B < C}

These restricted versions of *precedence* and *dominance*, defined only between adjacent elements, encodes in a very transparent way significant linguistic relations such as *merge* (Chomsky 1999): "A < B", in fact, corresponds to "A *merges* with B and *projects over* B" (namely A is the *head* of the constituent resulting from *merge*):

(4) $A < B = \text{merge}(A, B) \rightarrow [_A\ A\ B]$

On the other hand, the necessity to describe phrase structures also in terms of discontinuous constituency dependencies (chains/movements, binding relations, ellipses), can be captured within the revisited SD formalization given in (3), following the intuition that an element is "merged" in more than one position within the phrase structure, even though it is (fully) pronounced (then *phonetically linearized*) only in one of these positions (usually the structurally highest one). The interplay between *immediate dominance* and *immediate precedence* defined among lexical elements in the phrase structure can help us to capture this intuition:

(5) **Long Distance Dependencies** (definition)

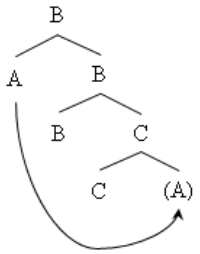
two non-empty elements enter a *long distance dependency* (thus forming a discontinuous constituency relation) when an *immediate dominance* relation but no *immediate precedence* relation is defined between them.

Note that a phonetically null element is not necessarily an empty element (since semantic and other formal features should be present). For instance given the following SD, A and C enter a Long Distance Dependency since C<A exists but neither <A,C> nor <C,A> is present:

(6) **I** = A, B, C
P = <A, B>, <B, C>
D = B < A, B < C, C < A

This SD can be graphically represented by the tree below:

(7)



The Long Distance Dependency reported in (6)–(7) is essentially an instance of *movement* (Chomsky 1995-2001) even though the directionality of the arrow is inverted to indicate that the highest element provides feature values for interpreting the underspecified features on the lowest "copy").

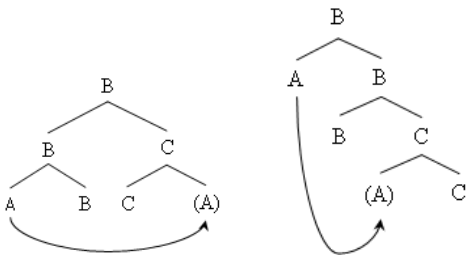
There are empirical reasons to believe that it would be possible to extend this approach to *pronominal binding* (on the line of Kayne 2002) and to *control* (Hornstein 1999, Boeckx, Hornstein 2004) capturing major typologies of Long Distance Dependency, even though I will not discuss the issue in these pages.[2] Note that a SD defined

[2] Again, refer to Chesi (2004) for a full discussion of the empirical scope of the proposal.

as in (3) with simply the information given in (6) is not able to discriminate among the right branching structure given in (7) and other structures such as the ones drawn below:

(8)

a. b.



Without using *assignment functions*, as proposed in (2)), we can rule out the unwanted representations by posing extra constraints on the occurrence of *long distance dependencies* and on the general shape of the SD. The *Linear Correspondence Axiom, (LCA)* (Kayne 1994), for instance, would suggest a deterministic mapping between *dominance* and *precedence* depending on the structural function of the lexical items involved in the SD strictly predicting a right branching structure. With the same spirit, I will adopt the following principle:

(9) **Linearization Principle** (inspired to *LCA*, Kayne 1994)
If A $<$ B, then either
a. $<$A, B$>$ if B is a *complement* of A (that is, A selects B), or
b. $<$B, A$>$ if B is a *functional projection*[3] of A

Following the notion of *merge* (in terms of *immediate dominance*) given in (3), we can define an useful asymmetric relation among the nodes within a tree:

(10) **Asymmetric C-Command** (definition)
When two elements *merge*, they respectively *asymmetrically C-command* all constituents of their sister

This definition is sufficient to discard (8.a) under the relatively standard constraint on movement provided below:

(11) **Constraint on movement** (definition)
A moved element always *asymmetrically C-Commands* its trace(s)

As it will be clear later on, we do not need to discard (8.b) since within this formalization the *linearization* of a trace is irrelevant.[4] Then (8.b) and (7) are equivalent.

[3] Assume *functional projection* to be synonym both of *specifier* and of *functional position*, following Starke (2002).
[4] But see Nunes (2004) for a different perspective.

1.2 Minimalist Grammars

We can think of a language, that is an infinite set of grammatical expressions each associated at least with a single grammatical SD, as a way of productively restricting the theoretically possible *precedence*/*dominance* relations that can co-occur within the same sentence. We refer to the intensional procedure that characterizes this set, as speaker's *competence*. Formally speaking, this *competence* can be described by a *grammar* that, following the minimalist trend, includes, at least, the specification of a *lexicon* (a finite set of *words* built from an *alphabet* with associated specific *features*) and a set of *Structure Building Operations*.

Stabler (1997) proposes a clean formalization of a *Minimalist Grammar* (*MG*) as outlined in Chomsky 1995 that includes these two basic components (*Lexicon* and *Structure Building Operations*). Following his proposal, a MG can be defined as a 4-tuple {V, Cat, Lex, F} such that:

(12) **Minimalist Grammar** (**MG**, Stabler 1997)

V is a finite set of non-syntactic features, $(\pi \cup \sigma)$ where π are phonetic features and σ are semantic ones;

Cat is a finite set of syntactic features, Cat = (*base* ∪ *select* ∪ *licensors* ∪ *licensees*) where

base are standard categories {complementizer, tense, verb, noun ...},

select specify one of the three possible kinds of selection {=x, =X, X= | x ∈ *base*} where =x means simple selection of an x phrase, =X selects an X phrase, suffixing the selecting head with the phonetic features of the selected X phrase; X= selects an X phrase, prefixing the selecting head with the phonetic features of the selected X phrase,

licensees specify requirements forcing phrasal movement {-wh, -case ...}, -x triggers covert movement, while -X triggers overt movement,

licensors are features that can satisfy licensee requirements {+wh, +case ...};

Lex is a finite set of expressions built from **V** and **Cat** (the *lexicon*);

F is a set of the two partial functions from tuples of expressions to expressions {*merge*, *move*};

The language defined by such a grammar is the closure of the *lexicon* (*Lex*) under the *structure building operations* (*F*). (13) is a simple example of MG able to deal with simple *wh*- movements[5]:

[5]For the sake of simplicity, let us assume that capital features directly *select* the position of the arguments without involving pre/in-fixing (then, =X means that the argument X is *selected* to the right of the selecting head, while X= to the left). The very same result is however derivable by a combination of standard (non directional) *selection* plus a trigger for *movement* (for instance -case) or assuming, as Stabler does, a difference in the merge result whenever *complex* or *simple* trees are merged.

(13) MG example

V = π = {/what/, /did/, /you/, /see/}, σ = {[what], [did], [you], [see]}

Cat = *base* = {D, N, V, T, C}, *select* = {=D, =N, =V, =T, =C}, *licensors* = {+wh}, *licensees* = {-wh}

Lex = [$_{-\mathrm{wh}\ \mathrm{D}}$ what], [$_{=\mathrm{V}\ \mathrm{T}}$ did], [$_{\mathrm{D}}$ you], [$_{=\mathrm{D}\ \mathrm{D}=\ \mathrm{V}}$ see], [$_{=\mathrm{T}\ +\mathrm{wh}\ \mathrm{C}}$]

F = *merge*, *move* such that:

merge (X, Y) = is a function taking two adjacent subtrees X and Y, outputting an unified structure Z of the form [$_{\mathrm{X}}$ X Y] if and only if X has as a first selecting feature (=f, =F, F=) and Y has the needed selected feature F as the first feature of the base set

move (X, Y) = if a function taking two subtrees [$_{+\mathrm{g}}$ X] and [$_{-\mathrm{g}}$ Y] such that <[$_{+\mathrm{g}}$ X], W, [$_{-\mathrm{g}}$ Y]> (where W can be any possible subtree, even null, but without any selecting/selector feature *g* in it) and produces Z of the form [[$_{\mathrm{X}}$ Y X] W, t_{Y}]

Following Chomsky, a derivation proceeds *from-bottom-to-top*[6] and *licensees* trigger *movement* as shown below[7]:

(14) **1.** *merge* ([$_{\mathbf{=D}\ \mathrm{D}=\ \mathrm{V}}$ see], [$_{-\mathrm{wh}}$ **D** what]) → [$_{\mathrm{see}\ \mathrm{D}=\ \mathrm{V}}$ see, $_{-\mathrm{wh}}$ what]

2. *merge* ([$_{\mathbf{D}}$ you], [$_{\mathbf{D}=\ \mathrm{V}}$ see, $_{-\mathrm{wh}}$ what]) →
[$_{\mathrm{see}}$ you, [$_{\mathrm{see}\ \mathrm{V}}$ see,$_{-\mathrm{wh}}$ what]]

3. *merge* ([$_{\mathbf{=V}\ \mathrm{T}}$ did], [$_{\mathrm{see}}$ you, [$_{\mathrm{see}}$ **V** see, $_{-\mathrm{wh}}$ what]]) →
([$_{\mathrm{did}\ \mathrm{T}}$ did, [$_{\mathrm{see}}$ you, [$_{\mathrm{see}}$ see, $_{-\mathrm{wh}}$ what]]]

4. *merge* ([$_{\mathbf{=T}\ +\mathrm{whC}}$], [$_{\mathrm{did}}$ **T** did, [$_{\mathrm{see}}$ you, [$_{\mathrm{see}}$ see, $_{-\mathrm{wh}}$ what]]]) →
([$_{\mathrm{C}\ +\mathrm{wh}\ \mathrm{C}}$, [$_{\mathrm{did}}$ did, [$_{\mathrm{see}}$ you, [$_{\mathrm{see}}$ see, $_{-\mathrm{wh}}$ what]]]])

5. *move* ([$_{\mathrm{C}}$ **+wh** $_{\mathrm{C}}$, [$_{\mathrm{did}}$ did, [$_{\mathrm{see}}$ you, [$_{\mathrm{see}}$ see, **-wh** what]]]] →
[$_{\mathrm{C}}$ What, $_{\mathrm{C}}$, [$_{\mathrm{did}}$ did, [$_{\mathrm{see}}$ you, [$_{\mathrm{see}}$ see, t_{what}]]]]

Some interesting formal results show that there is a weakly equivalent *Multiple Context-Free Grammar* for any MG (then MG are included in the *Mildly Context-Sensitive* class of grammars, Michaelis 1998) and that a *recognizer* algorithm can be defined (both top-down and bottom-up) for MGs (Harkema 01). However, it is difficult to draw any computational/cognitive conclusion from these results, because either they are based on a *deductive parsing perspective* (Shieber and al. 1995) or on a *weak equivalence* between MGs and other formalisms (e.g. *Multiple Context-Free Grammars*): namely, these grammatical formalisms/derivations can produce the very same set of strings MGs will produce, but either they fail to associate (derivationally) the same structures to these strings or they encode *lexical items*, *features* and *structure*

[6]That is, from the inner verbal head, adding piecemeal *complements* then *functional specifications*, up to the most external heads.

[7]This is a very simplified version of derivation, to be taken only as example. It would be clearly possible including the subject movement too, but this would have been required extra steps in the derivation.

building operations in a less transparent way with respect to the linguistic intuitions that justified them. I believe that these two factors are indeed crucial parameters of evaluation, at least if the final goal of the formalization is that of making clear what the computational (and possibly psycholinguistic) implications are both in *parsing* and in *generation* if we want to use MGs effectively in these two contexts. A fundamental step is then to define precisely how the *parsing* and the *generation* tasks could be described in order to pose some effective "interface conditions" on the grammar formalism.

1.3 Two (sub-)problems for parsing and generation: ambiguity and long distance dependencies

Given a MG *G* defined as in §1.2 and assuming that a *Structural Description* (*SD*) can be expressed basically in terms of *immediate dominance* (*D*) and *immediate precedence* (*P*) as proposed in §1.1, we can define the *parsing* and the *generation* tasks (symmetrically) as follows:

(15) **The parsing problem** (definition)

given a finite set of *phonetic features* π (grouped by words) and a *precedence* total order among them, find the relevant set of lexical items *Lex*, compatible with π and the set of *(immediate) dominance* relations *D* among σ features associated to π in *Lex*, if possible, if not reject the input.

(16) **The generation problem** (definition)

given a finite set of *semantic features* σ and a finite set of *dominance* relations *D* among them, find the relevant set of lexical items *Lex* and the correct *linearization* among π features associated to σ in *Lex*, if possible, if not reject the input.

Both problems can be factored in two distinct sub-problems: a part of *lexical ambiguity resolution* (... "find the relevant set of lexical items Lex"... compatible with π/σ...) and a part of *structural mapping* (from *precedence* to *dominance* and vice versa). These problems are difficult to solve, because of a non-univocal mapping (*non-determinism*) between phonetic features and words (homophony/homography, polysemy etc.) and because of *discontinuous constituencies* (*long distance dependencies*) which, among other factors such as *PP attachment* and *constituents conjunction*, cause structural ambiguity. I assume that these two problems pose interesting constraints (much less abstract than usual "interface conditions", Chomsky 1995) on the grammar specification if we accept the intuitive idea that the very same *competence* (then the very same *grammar*) has to be used in both contexts. As it will be clear in the next paragraph, *Structure Building Operations* are especially affected by these considerations.

1.4 Structure Building Operations and directionality

There are empirical and formal reasons (Chomsky 1995-2001) to believe that a minimal specification of the grammar can include *Structure Building Operations* such as

merge and *move* (**F** in Stabler's MGs, §1.2). This seems to be plausible also from a performance perspective when we have to map *precedence* and *dominance* with respect to lexicalized feature structures both in *parsing* and in *generation* contexts according to the definition provided in §1.3. There are however theoretical and psycholinguistic reasons to believe that the *Bottom-to-Top* orientation of the derivation cannot be pursued neither from a *parsing* nor from a *generation* perspective: Phillips (1996), for instance, shows how *intermediate constituencies* built from *Top-to-Bottom/Left-to-Right* can account for important contradictory constituency tests (that is, when *movement* would predict a different constituency with respect to *scope relations* in sentence such as "John gave a candy to any children in the library on Sunday..."); on the other hand, the cyclic idea of movement (*move short*, Chomsky 1995) and the notion of *derivation by phase*, require purely formal *uninterpretable features* to trigger intermediate steps in any *long distance movement* context. This seems to be a non-deterministic, unsatisfactory solution and it is definitely not suitable from a *parsing* perspective.[8] Moreover psycholinguistic evidences show that both in *generation* (*false starts*) and in *parsing* (*garden paths*) the orientation of the processing could fairly be from-left-to-right/top-to-bottom; last but not least, as far as I know, no clear psycholinguistic evidence supports the bottom-to-top model.

Taking into account these cues[9], the definitions of SDs (§1.1) and MGs (§1.2), assume that any item is licensed within a SD if and only if it is *selected*[10] or it is a plausible *functional specification*[11] of a lexical head according to the *Linearization Principle*. Assume, furthermore, that *merge* could be defined as a binary function which takes two feature structures and unifies them (in the sense of *unification grammars*, Shieber 1986).

In order to account for the above mentioned problems, I will re-define *move* as a top-to-bottom/left-to-right oriented function which stores an *unselected* element in a sort of *memory buffer* and *re-merges* it at the point of the computation where the element is *selected* by a lexical head (according to the *select* features of this lexical head and to the *Linearization Principle*).[12] This modification would allow us to get rid of the *uninterpretable features* hypothesis, then removing the teleological behavior behind the movement conception as a feature driven, bottom-to-top operation.

Beyond the empirical adequacy of these redefinitions of the *Structure Building Operations merge* and *move*, which I will not evaluate in these pages (as explained in fn.1), it would be necessary to evaluate their computational impact. In fact, while the *merge* operations has a local scope and does not present dramatic complexity issues (at

[8] Neither the notion of *numeration* makes any clear sense in *parsing*, nor the idea of adding arbitrary *uninterpretable features* to the lexical items in order to "reconstruct" a movement operation.

[9] For more datails refer to Chesi (2004).

[10] From this perspective selection means both *C(ategorial)-selection* (then it operates on *Cat* features) and *S(emantic)-selection* (Pesetsky 1982), then it operates on *semantic* features).

[11] A sort of *licensor* specification in Stabler's terms. According to *Cartography* (Cinque 1999–2002) I assume an articulated *functional structure* above any lexical elements, moreover expecting that the legitimated position within the structure has to be evaluated in terms of relative scope of the items effectively present within the SD.

[12] The simplest possible device to store un-selected elements out of the SD would be a stack, First-In-First-Out memory; this simple proposal is able to account for argumental movement, topicalization and wh-movement. See Chesi (2004) for solutions with wider empirical coverage.

least once the feature set of the lexical items are clear) the long distance nature of the *move* operation seems to be allegedly onerous if not bounded in its potentiality: capturing arbitrarily distant relations would not be a welcome result neither empirically nor psycholinguistically; moreover it could be computationally very expensive.

Introducing the idea of *derivation by phase* seems to be clearly possible also in this Top-to-Bottom oriented framework and it could be the solution for many problems roughly introduced before: from this perspective, a *phase* can be thought as the minimal part of a Top-to-Bottom computational process in which all the *functional* and *selectional* specifications associated to a given lexical head (i.e. N(oun) or V(erb)) are satisfied. Crucially, a *phase* can be included within F as a "Top-to-Bottom expectation" (*phase projection*), as follows:

(17) **Phase Projection** (definition)

a projection of the minimal set of *D* relations, optionally underspecified for π or σ features, within the *SD* according to the *select* features present in the processed *lexical head*.

Note that in order to make this device computationally interesting we should distinguish between "active" and "inactive" phases: informally speaking, a phase is active or *open* when their elements can still enter new *merge* relations (either because the head of the phase has not been processed yet or because of *move* which stored some of these elements within the memory buffer); on the other hand, a phase become inactive or *closed* when any required relation has been established: once processed the (lexical) head of the phase, *Phase Projection* applies, according to the *select* features present in the phase head, and the phase is closed; any element still present in the memory buffer at this point is inherited (remerged) within the lower projected phases.[13]

Rephrasing these notions of phase in more formal terms, given a grammar $G = \{V, Cat, Lex, F\}$, an input i to be processed, composed by elements belonging to *Lex* (even if specified only for π or σ features), considers:

(18) **Phase** (definition)

a complete process (of *parsing* or *generation*) involving:

- a *Phase Projection* about a potential SD structure,
- a proper subset i_{p} of i such that any item in i_{p} is a lexical element (*N* or *V*), the *head of the phase*, or it is a *functional specification* of the head of the phase,
- a memory buffer *M* to store unselected items and retrieve selected ones, initialized as empty, unless some items are inherited from the non-empty memory buffer of a selecting previous phase (this inherited element will be part of i_{p} and will be legitimated within the phase by merging it at the relevant position of the left-periphery, edge of the phase, as phonologically null element, unless otherwise specified);

[13] In fact the whole story would be a bit more complex. For sake of simplicity this is however enough to understand the following discussion. See Bianchi, Chesi (2005) for a detailed proposal on the distinction between *nested* and *sequential phases*.

(19) **Phase Completeness** (definition)

a phase is complete (i.e. *closed*) iff the head of the phase is saturated, namely if any mandatory selectional requirement (direct object, indirect object etc.) is satisfied (a *Phase Projection* can be considered a complete phase in order to close the phase generating this top-to-bottom expectation);

(20) **Phase Complementation** (definition)

a phase takes only *complete phases* as complements;

(21) **Phase Selection Requirement** (definition)

a phase has to be *selected*; items in the memory buffer, at the end of the phase, can be transferred only to the memory buffer of the selected phase(s).

The goal of using *phases* is then to reduce the potential non-determinism of *ambiguities* and *long distance dependencies resolution* both restricting the space of the problem in an empirically adequate way and forcing the algorithm to bias its choices following precise Top-to-Bottom expectations (that is, a precise set of *immediate dominance* relations underspecified for some features).

2 Reducing Complexity using phases

The complexity of a problem is an expression of the resources (essentially *time* and *memory*) needed to solve the problem (Papadimitriou 1994). More precisely, it is a function of the size of the problem, determined at least by three factors:

(22) **a.** the length of the input (n);

b. the space of the problem (all states the system can attaint by correctly applying any legal rule);

c. the algorithm used to explore this space.

While a. is largely independent from the grammar, b. and c. are strictly determined by the linguistic theory. From a MG perspective, b. is mostly determined by the *lexicon* and by the *structure building operations merge* and *move*; in the *top-to-bottom* perspective just mentioned, c. is a procedure driven by the *Linearization Principle* that has to inspect items and recognize which one is *licensed* in a specific position and which one has to be *moved* and *re-merged* in another (lower) position. Let us explore the complexity of the two (sub)problems of *ambiguity* and *Long Distance Dependency* identification with respect to these three parameters.

2.1 The complexity of Ambiguity

Considering both lexical and semantic ambiguity the first (sub)problem can be stated as follows:

(23) **Ambiguity problem** (definition):

given an input *i*, composed by π (in *parsing*) or σ (in *generation*) features grouped by words, of length *n*, and a number *c* of possible *Part-of-Speech* (*PoS*)[14], assign to each word from the input at least one *PoS*, if possible, if not reject the input.

The *complexity* of the problem, considering a brute force algorithm to solve it, is at worse O(c^n) (assuming that any word could be ambiguous among all *PoS*). A more realistic complexity order can be guessed by inspecting *dictionaries* or *corpora*. Using *Wordnet* (Miller 1995), the ambiguity factor[15] would decrease down to about 1.44 (this factor seems to be fairly steady across languages such as English, Spanish and Italian), then we would obtain an order of complexity of O($1,44^n$). Slightly more optimistic results can be obtained with the *Brown corpus*: 40% of the words seem to be ambiguous and most of them are ambiguous between two *PoS*; then we could approximate the ambiguity factor up to 40%, obtaining an order of complexity of O($1,4^n$). This is clearly not enough yet for the problem to be tractable: analyzing a text would imply processing hundreds of words;[16] the exponential combinatory of the problem makes it impossible to find out a plausible solution in an acceptable time. Then, we should restrict the combinatorial domain across the input and/or use plausible clues to restrict the range of *ambiguity*.

One possible move is to consider the domain of *ambiguity resolution* to be restricted to a limited context: in fact, if the exponential *n* in the complexity function turns out to be a fixed number, the problem becomes tractable. But of course the context cannot be arbitrarily fixed (e.g. *n-grams* approach): the length of a grammatical phrase containing ambiguities can be arbitrarily long (theoretically infinite exploiting the complementation option), then fixing it once and for all would not be heuristic. It is also implausible to reduce the "structurally defined" context to the simple *local selection* as shown by the following contrasts:

(24) a. The [dogs] run ([D the] selects [N dogs])

b. Mary [dogs] me ([DP Mary] does not select any [V dogs])

The very same problem rises with *adverbials selection*: an adverb cannot *select* (in a technical sense) all the possible lower adverbials (that obviously can be present or not without affecting the grammaticality of the sentence). A more adequate solution could be to define the *phase* as the "largest context" within which an ambiguity can be solved: this would imply using the set of *dominance* relations projected according to *phase projection* to reduce the number of possible *PoS* associated to the set of *phonetic/semantic* features. Since a *phase* is dimensionally bounded in length (maximum number of *precedence* relations) and in depth (maximum number of *dominance*

[14] *Categories* (or *PoS*) have to be intended as "indices" (*Synsets* assuming the WordNet terminology) pointing to fully specified (in terms of features) items in the lexicon.

[15] ambiguity factor = $\frac{\text{synsets}}{\text{lexicalentries}}$

[16] With just 50 words to be disambiguated we could evaluate up to about 20 millions of possibilities.

relations)[17], the complexity order within each *phase* would be O($1, 4^{k+1}$) (where *1,4* is the most realistic ambiguity factor based on *dictionaries* and on *corpora*): the fundamental detail in this *complexity function* is that the exponent is, this time, a fixed number, virtually independent of the length *n* of the input. Note that opening more than one *phase* (from this perspective, any unselected argument, e.g. *preverbal subject*, represent a new phase opened within another *phase*) would produce an increase of the complexity order of O($1, 4^{p(k+1)}$) where *p*, the number of open *phases*, could grow boundlessly, in principle, leading quickly to intractability. This is however a welcome result, since a degradation of the human linguistic *performance* is reported in many processing data when the subjects have to parse/generate sentences with certain ambiguous structures that clearly show a difficulty in "keeping unsolved ambiguities" (e.g. "buffalo" sentences). Thus, the more *phases* we open (at the same time) the more difficult the problem will be as empirically expected.

2.2 The complexity of Long Distance Dependencies

Considering a *parsing* perspective[18] and a brute force algorithm, finding out which *dominance* relations have to be associated to a given set of *precedence* relations given in input has, at least, the complexity order of O(2^{n-1}), where *n* is the length of the input (namely the number of words in the sentence): this is because among *n* items we should define *n-1* relations at best (the minimum number of relations that would make a tree of *n* leafs, fully connected) and any of these relations can be ambiguous about the projecting head (A<B or B<A). Complexity rises, if we consider simple *movement* (let us put aside for the moment *cyclic movement*): at worse any item could have been potentially moved out from any lower (*C-commanded*, *selected*) position, the complexity order of the problem increases up to O($2^{(n^2-n)/2}$): this is because, potentially, any element could establish a dominance relation with any other element that follows it: e.g. with *4* elements {A, B, C, D} we could have *6* possible dominance relations {A-B, A-C, A-D, B-C, B-D, C-D}; with *5* elements {A, B, C, D, E} we could have *10* possible dominance relations {A-B, A-C, A-D, A-E, B-C, B-D, B-E, C-D, C-E, D-E}... then $\frac{((n-1)+1)\times(n-1)}{2}$. Complexity rises again (boundlessly this time), considering that empty heads[19] can enter *dominance* relations and there is no way to determine how many empty heads could be present, in principle, in a sentence of length *n*.

This is clearly not a satisfactory result, since the growing rate of any of these func-

[17] Given a fixed hierarchy of *licensors* features (Cinque 1999–2002), a *phase* contains at worst **k** *functional elements* (or *licensors*), exactly **1** projecting lexical element (by definition of *phase*). Remember then, that by assumption (Pesetsky 1982) each lexical head selects at most **3** *ph(r)ases* (subject, object and indirect object); this determines the maximum growth of the progression composed by the number of the (selected) phases which will be projected.

[18] I will not consider in these pages the *generation* problem, which is however much more "easy" than the *parsing* one: linearizing a set of dominance relations in a phase by phase procedure is straightforward once we have the *Linearization Principle* and a rigid order predicted among licensors features. A non trivial problem to be discussed is how to retrieve the exact set of dominance relations which belong to the phase, but this very same problem affects Chomsky's *numeration* idea too.

[19] At least in terms of π features.

tions would make the problem quickly intractable. Once again intractability in *parsing* can be tacked by adopting the idea of *phase* then working out the (22.c) factor: we assumed before (§1.4) that *movement* can be detected by the presence of an element that is not *selected* (that is, an element which is not selected by a previously processed lexical element, according to the *Linearization Principle*, (9)); in this case, this element would stand in "memory" as long as another element *selects* it.

Following Chomsky (1999), let us assume that if *movement* happens, it has to happen within the *phase*: then given a limited context corresponding to the *phase* boundaries, either we find the *selecting* element within it, so we can connect the moved object to its base position (projecting a set of *dominance* relations, according to *Phase Projection*, (17), that minimally satisfy this *selection* necessity), or we do not find any *selecting* element, then the expectations to find a *selecting* sister for the unselected element will be projected onto the *lower (selected) phase*[20] and so on. The properties of the intermediate traces[21] strengthen the idea that these traces on the *edge* (Chomsky 1999) of the *phase* serve as a sort of "memory refresh" (*phase balance*, Felser 2001), that is, the undischarged elements within the previous *memory buffer* are allowed to enter the "phase numeration" (then being part of the next selected *phase* processing) in order to make *Long Distance Dependencies* possible. This intuition produces a remarkable reduction of *complexity* (as shown in table 1): in fact, for any *moved* element, we would have only two possible landing site positions within a *phase* to be evaluated (one *left-edge* position, used as a memory refresh or one *selected* position). Then for any *phase* the number of *dominance* relations required would be, at worst, $2^{2k)}$ (in case anything would have been moved to the licensors field of the *phase*).[22] The complexity order of the problem, considering any *dominance* as ambiguous, would be O(2^{2k}). Opening a new *phase* before having closed the previous one, again, leads to a duplication of the number of possible *dominance* relations and so on as long as new *phases* are opened. The real order of the problem is then O(2^{p2k}) with *p* representing the number of open *phases* at the same time. The relation between the *length of the input* (*n*) and this function is expressed in terms of *phases*, since any *phase* represents a chunk of this input; in particular, the number of *lexical items* in *n* would determine the number of *phases* (which could be *n*, at worst). Note that *Discontinuous Constituency Dependencies* within a *phase* would not produce any remarkable effect on the *complexity* of the problem, while *discontinuous constituency relations* among *phases* would increase the *complexity* of the problem in a linear way if any *phase* is closed before the next one starts. On the other hand, there is an exponential increase of *complexity* any time we open a *phase* before having closed the previous one.

This expected increase of the *complexity* (parallel to the degradation in processing)

[20] Leaving an intermediate trace on the "left-periphery" of this lower *phase*, as discussed in Chesi (2004), Bianchi, Chesi (2005).

[21] They are not *selected* positions; they are available for *binding*, *reconstructions effects*; they are not triggered by any apparent satisfaction of semantic requirements, they are in fact accounted for, in a *bottom-to-top* perspective, by purely formal features.

[22] **k** is the number of functional features present in the current phase (which potentially can legitimate movement to their specific position), **1** lexical head, minus **1**, provide that for linking *n* elements we would need *n-1* relations.

functions	N° *of relations* *n*=6 (*p*=2, *k*=2)	*to be evaluated* *n*=9 (*p*=3, *k*=2)
Brute force $2^{(n^2-n)/2}$	$\simeq$ 32K	$\simeq$ 68.000M
Nested Phases 2^{p2k}	256	4096
Linear Phases $p \cdot 2^{2k}$	32	64

Table 1: Comparison among functions w.r.t. input length (assume each phase to be composed by 2 licensors features/positions and 1 lexical head).

related to the *movement* of elements across *open phases* exactly predicts *center embedding effects* (Chomsky 1965) and *strong island conditions* (Huang 1982, Bianchi, Chesi 2005).

3 Conclusion

In this paper I proposed a formalization of the notion of *phase* (Chomsky 1999) providing some arguments in favor of the necessity of including this component as part of the *Structure Building Operations* within a *Minimalist Grammar* formalism based on Stabler's 1997 proposal. Moreover, I assumed that the formalization of the other *Structure Building Operations merge* and *move*, also has to differ from Stabler 1997 in terms of *directionality*: the standard *Bottom-to-Top* derivation assumed in Chomsky's work has been show to be problematic from many perspectives (Chesi 2004) then it has be replaced by a *Top-to-Bottom*, *Left-to-Right* perspective, extending Phillips' 1996 original approach. Within this perspective, formalizing performance tasks such as (*parsing* and *generation*) has been shown to be a very useful tool in order to highlight the essential parameters to calculate the complexity function of a generalized algorithm that has to make an effective use of this *competence model*, pointing out that the formalized notion of *phase* and the directionality of the *move* operation produce a (psycholinguistically plausible) remarkable complexity reduction in both *ambiguities* and *long distance dependencies* resolution.

References

Bianchi, V. and Chesi, C.(2005), Phases, left branch islands and computational nesting, *Proceedings of the 29th PENN LINGUISTICS COLLOQUIUM*.

Boeckx, C. and Hornstein, N.(2004), Movement under control, *Linguistic Inquiry* **35**(3), 431–452.

Chesi, C.(2004), *Phases and Cartography in Linguistic Computation: toward a Cognitively Motivated Computational Model of Linguistic Competence*, PhD thesis, University of Siena.

Chomsky, N.(1995), *The Minimalist Program*, MIT Press, Cambridge (MA).

Chomsky, N.(1999), *Derivation by Phase*, MIT Occasional Papers in Linguistics, no. 18. Cambridge (MA).

Chomsky, N.(2001), *Beyond Explanatory Adequacy*, MIT Occasional Papers in Linguistics, no. 20. Cambridge (MA).

Cinque, G.(1999), *Adverbs and Functional Heads: A Cross-linguistic Perspective*, Oxford University Press.

Cinque, G.(2002), *The Structure of DP and IP. The Cartography of Syntactic Structures, Vol.1*, Oxford University Press.

Felser, C.(2001), Wh-copying, phases and successive cyclicity, *Essex Research Reports in Linguistics*.

Harkema, H.(2001), *Parsing Minimalist Languages*, PhD thesis, UCLA.

Huang, J.(1982), *Logical relations in Chinese and the theory of grammar*, PhD thesis, MIT.

Kayne, R.(1994), *The Antisymmetry of Syntax*, MIT Press, Cambridge (MA).

Kayne, R.(2002), *Pronouns and their antecedents*, Blackwell, Oxford.

Michaelis., J.(1998), *Derivational Minimalism is Mildly Context-Sensitive*, Springer Verlag, Berlin.

Miller, G. A.(1995), Wordnet: A lexical database for english, *Communications of the ACM*.

Nunes, J.(2004), *Linearization of Chains and Sideward Movement*, MIT Press, Cambridge (MA).

Papadimitriou, C.(1994), *Computational Complexity*, Addison-Wesley, Reading (MA).

Pesetsky, D.(1982), *Paths and Categories*, PhD thesis, MIT.

Phillips, C.(1996), *Order and Structure*, PhD thesis, MIT.

Shieber, S., Schabes, Y. and Pereira, F.(1995), Principles and implementation of deductive parsing, *Journal of Logic Programming*.

Stabler, E.(1997), Derivational minimalism, *Lecture Notes in Computer Science* **1328**, 68–98.

Starke, M.(2002), Against specifiers, *Abstract for TILT 2002*.