COMPETENCE AND COMPUTATION:
TOWARD A PROCESSING FRIENDLY MINIMALIST GRAMMAR

by

Cristiano Chesi
Cristano Chesi

*Competence and Computation:*
*toward a processing friendly minimalist Grammar*

© 2012 by UNIPRESS
via Venezia 4/A – 35131 PADOVA, Italy
www.unipress.it
all rights reserved

## CONTENT

<table>
<thead>
<tr>
<th>Acknowledgments</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>7</td>
</tr>
<tr>
<td>- Chapter 1 - On cognitive plausibility of the Linguistic Computation</td>
<td>13</td>
</tr>
<tr>
<td>1.1 What’s a Feature</td>
<td>14</td>
</tr>
<tr>
<td>1.1.1 Features in linguistics</td>
<td>17</td>
</tr>
<tr>
<td>1.1.2 Feature typologies</td>
<td>20</td>
</tr>
<tr>
<td>1.1.3 Features from the lexicon: the inclusiveness condition</td>
<td>24</td>
</tr>
<tr>
<td>1.1.4 Feature hierarchies</td>
<td>31</td>
</tr>
<tr>
<td>1.1.5 Categorial features</td>
<td>38</td>
</tr>
<tr>
<td>1.1.6 Features structures: x-bar theory and functional sequences</td>
<td>42</td>
</tr>
<tr>
<td>1.1.7 Features parameterization: how to account for variation</td>
<td>46</td>
</tr>
<tr>
<td>1.2 Basic Operations on Features</td>
<td>49</td>
</tr>
<tr>
<td>1.2.1 Merge as unification</td>
<td>51</td>
</tr>
<tr>
<td>1.2.2 Non-local dependencies: move and merge again</td>
<td>58</td>
</tr>
<tr>
<td>1.3 Relevant Relations among Elements: Introduction to Derivations</td>
<td>62</td>
</tr>
<tr>
<td>1.4 Complexity Theory</td>
<td>65</td>
</tr>
<tr>
<td>- Chapter 2 - Linguistic and Computational Models</td>
<td>71</td>
</tr>
<tr>
<td>2.1 Between competence and performance: processing models</td>
<td>71</td>
</tr>
<tr>
<td>2.2 Derivations or representations? some linguistic models</td>
<td>78</td>
</tr>
<tr>
<td>2.2.1 Extended Standard Theory (EST) and Government and Binding (GB)</td>
<td>85</td>
</tr>
</tbody>
</table>
2.2.2 The Minimalist Program

2.2.3 Left to Right Incremental Processing

2.2.4 The Cartographic approach and Locality Constraints on Movement

2.3 Computational Models

2.3.1 Principle-based Parsing

2.3.2 Minimalist formalization

2.3.3 Minimalist Parsing

- Chapter 3 - Computable Models of Competence

3.1 Aspects of the Grammar to be formalized

3.1.1 The bidimensional nature of the grammar

3.1.2 performance tasks

3.2 Two (computationally hard) linguistic phenomena: ambiguity and non-local dependencies

3.2.1 Ambiguity

3.2.2 Typologies of Non-Local Dependencies

3.3 Empirical inadequacies: re-defining Merge and Move

3.3.1 Cartography, Extended Projections and Phases

3.3.2 Merge reduces to Lexical Insertion

3.3.3 Directionality of Movement and Relativized Minimality

3.3.4 Move is triggered by unsatisfied local dependency

3.4 Complexity Issues
3.4.1 The Complexity of Ambiguity 156
3.4.2 The Complexity of Non-Local Dependencies 159
3.4.3 How Phases constraint Non-Local Dependencies 166

Chapter 4 - Grammar formalization in use 171
4.1 Parsing and Generating sentences using the same Grammar 172
4.2 Empirical coverage, the case of Strong Island 177
4.3 Concluding Remarks: Implementing Structure Building Operations 185

References 187
ACKNOWLEDGMENTS

After many years, this book finally came to an end. The book, not the grammatical theory here discussed that is more and more vibrant and promising than ever! Most of the merit of this enterprise should be endorsed to Valentina Bianchi who supported me during these years, before and after a preliminary version of this work became my PhD thesis under the supervision of Luigi Rizzi (University of Siena) and Robert C. Berwick (MIT, Cambridge MA). She always prompted me with clever hints to be formalized, juicy literature to be explored, and intricate empirical data to be explained, substantially contributing to the ideas and to the empirical coverage of the grammatical model here discussed. Without her, I will surely have dropped this venture a long ago. I wish to thank the CISCL (the Center for Cognitive Studies of Language at University of Siena) that hosted me and this research program on Phase-based, Top-Down Minimalist Grammars, especially Adriana Belletti and Luigi Rizzi, who always supported my work and encouraged me to pursue the computational linguistic path. Thanks also to the NeTS, the Center for Neurolinguistics and Theoretical Syntax at IUSS, Pavia, directed by Andrea Moro, where my research has been recently conveyed, and where I have found time and energy to “complete” this work.
INTRODUCTION

In these pages I will present a computational model of linguistic competence that aims at being cognitively motivated and mathematically sound. Considering the linguistic competence as the “speaker/hearer’s knowledge of his/her language” (Chomsky 1965:4), I assume this to be expressed by a (generative) grammar: a formal description of any possible well-formed sentence of a given language associated with at least one Structural Description (SD) that expresses how this sentence is understood/produced by the (ideal) hearer/speaker. Then the linguistic competence is an intensional (mechanic) procedure that characterizes an infinite set of well-formed sentences. Even though a generative grammar can be completely abstract, that is, independent from how the linguistic competence is really used by the (actual) hearer/speaker (cf. Chomsky 1995), it makes sense asking, both from a cognitive and from a computational perspective, whether or not this knowledge representation (or at least part of it) is suitable to be used both in comprehension, that is, when we perceive a sentence, and in production, namely when we generate it.

I will refer to this property as the flexibility of the grammatical formalisms:

I. *flexibility* (definition)

   a grammatical formalism is *flexible* if it can be directly used both in comprehension and in production.

---

1 To be intended as opposite to extensional procedure, namely as opposed to the exhaustive list of elements (i.e. well-formed sentences) of this set.

2 Do not confuse *flexible* with *reversible* (e.g. reversible natural language grammars, Neumann 1994): *reversibility* is a mathematical property not necessarily implied by this informal notion of *flexibility*.
This is an important guiding hypothesis, necessary not only for developing computational linguistic resources (such as lexica and grammars) that can be effectively used in performance tasks, but also for underlining aspects that can be evaluated from a processing/cognitive perspective.

Beyond flexibility, three more properties confer to the grammar a cognitive plausibility:

II. **universality** (descriptive adequacy)
    a grammar should capture linguistic generalizations on empirical ground, cross-linguistic similarities, and account for variations;

III. **explanatory adequacy**
    a grammar should be learnable by a speaker/hearer;

IV. **realism**
    once established what performance is about, grammar + performance should faithfully reproduce productive phenomena involved in comprehension and production; the computational model that describes this facts should be tractable.

These properties pose many constraints at the formal level: to my knowledge no linguistic theory satisfies all these properties at the same time. Most of the frameworks, in fact, remain agnostic on some of these issues, causing difficulties for implementing a computational model that aims to be flexible, universal, learnable and realistic.

For instance, much work has been done with respect to the universality and learnability issues within the “generative tradition” (Chomsky 1957-2008, cf. §2.2). Even though many (radically) different perspectives have been explored in the last sixty years, flexibility and realism have always been rare topics.

Notably, the Minimalist Program represents an interesting attempt to reach an adequate descriptive adequacy dealing with complexity issues and phrase structure
building procedures in a derivational (that is mainly incremental) way. Even though the early warning on “abstractness” is still valid, this can be considered as a legitimate move toward a cognitively more plausible model of grammatical description.

In particular, the Program tried to reduce significantly the core grammatical inventory to simple (generalized) operations that concatenate lexical items: Merge, Move and Agree, where Merge is the fundamental operation for building objects: it takes two adjacent elements \( \alpha \) and \( \beta \) and combines them together creating a new object \( \gamma \) minimally of the form \{\( \alpha \), \( \beta \)\}; Move expresses the relation between two not adjacent objects (e.g. \( \alpha \) and an empty position \( t_\alpha \) (a trace) to be interpreted as an exact copy of \( \alpha \)). Agree, as well, forms a long distance relation between two non-adjacent elements, but it does not require movement and it is somehow more free.

As noted by Starke (Starke 2001) Merge and Move are both subject to strict constraints, also known as locality conditions (Rizzi 1990): Merge requires strict adjacency (an element can only merge with another element that is next to it), while Move creates long distance relationships, unless an intervening element of the same structural type (in a sense that will be explored in details in §2) is present between the object and the invisible/incomplete copy of the object.

Despite some relevant attempts to clarify the formal part behind the Merge and Move intuitions (Stabler 1997 and related work) many problems remains to be explored:

- many primitives remain underspecified; among them, crucially, the organization of the lexicon; the exact inventory of interface conditions; the format of the parameters; the conditions on feature checking;

- it is hard to find an internal coherence also among the small set of formal devices included (Merge, Move, Agree and the notion of Phase); in order to make a model consistent, the principles/rules used should be non-contradictory.
Often, when the set of “axioms” is huge (e.g. Extended Standard Theory, §2.2.1) and extracted from different analyses which solve local empirical problems, contradiction is an emergent property, hence difficult to be detected; the computational model described could encounter serious complexity and tractability issues; building a manageable model requires keeping in mind that the most economical assumptions are better candidates, where economical has to be intended both in a non-technical sense (simpler solutions have to be considered first, since they can be better evaluated/implemented; in this sense “reductionism” without loss of explanatory adequacy is an intriguing and necessary exercise we should deal with) and in a computational sense: some solutions are less expensive than others in terms of use of computational resources (basically time and memory, cf. Barton et al. 1987); minimalism does not present in any evident way a formalization of the devices proposed in order to reduce the overall complexity of the system even though this is a crucial point (for instance there is no accord on the notion of Phase, cf. §2.2.2, §3.4).

These are the rationales behind the choice of many scholars who have preferred adopting less powerful grammars (such as simple Regular Expressions or Context Free Grammars) for which we know efficient algorithms (for instance in CFG parsing: Earley, Earley 1970, and CYK algorithm, Kasami 1965, Younger 1967), completely redefining linguistic frameworks, formalizing/implementing new grammars that better fit with these “special” requirements (precise formalization, flexibility and realism; cf. Head-driven Phrase Structure Grammars, Pollard and Sag 1994, Tree Adjoining Grammars, Joshi 1985, Lexical Functional Grammars, Bresnan 2001).

In this book, I will propose that most of these difficulties can be overcome and that a computational description of some recent linguistic intuitions is indeed pursuable in order to reconcile empirical adequacy (universality and explanatory adequacy) with
a wider cognitive perspective (flexibility and realism). With the intent of doing that, in the first chapter of this book I will introduce the linguistic concepts that are used in most frameworks, emphasizing their cognitive nature (essentially comparing processing at other cognitive levels, such as vision); in particular, consideration will be given to defining what features are (§1.1), how they combine (§1.2) and which relations shall be defined among them (§1.3); additionally some essential concepts will be provided in order to understand cost functions (that are measures of the complexity) of the proposed devices (§1.4).

Generative frameworks (briefly mentioned in this introduction) will be systematically reviewed in the first part of chapter 2 (Principle and Parameters and the Extended Standard Theory, §2.2.1, the Minimalist Program, §2.2.2, and a “unusual” processing perspective for this framework, that is Phillip’s (1996) left-to-right processing model, §2.2.3; finally the Cartographical Approach will be explored in §2.2.4). The second part of this chapter will show some (partially successful) attempts to implement/formalize these linguistic frameworks (the Principle-based parsing approach, §2.3.1, Stabler’s minimalist grammar formalization, §2.3.2, and Fong’s minimalist parser, §2.3.3).

The rest of the book will try to solve some problematic issues: in order to provide a precise context, in chapter 3, I will firstly formalize the idea of structural description (§3.1.1) then the (performance) tasks that must access the grammar (essentially parsing and generation, which are specific cases of comprehension and production respectively, §3.1.2). Afterwards, the most problematic aspects of the language (ambiguities, §3.2.1, and long distance dependencies, §3.2.2) will be formalized with respect to the proposed performance tasks. Finally the formalization of a minimalist grammar (inspired by Stabler’s 1997 work and the Minimalist Program) enriched with considerations on the articulate geometry of the functional feature structures proposed within the Cartographic Approach will be discussed (§3.3).
From the theoretical perspective, the standard minimalist approach sketches a model that does not fit in a clear way with specific performance algorithms such as parsing or generation even though much emphasis is put on “interface properties”. The grammar formalized in §3.3 considers some important constraints posed both by the generation and parsing problems, in the end defining a model that should be:

1. usable both in parsing and in generation;
2. cognitively motivated;
3. tractable;
4. as much as possible deterministic.

To achieve these results, I will focus on three important properties that are not only cognitively plausible, but also formally and computationally advantageous (§3.4):

- structure building operations can be embedded within the grammar if they apply both top-down and left-right;
- using a Linearization Principle (inspired by Kayne’s 1994 LCA) and fixing the functional structure by mean of a universal hierarchy (Cartographic Approach) makes the derivation mostly deterministic;
- formalizing phases (cf. Chomsky 1999, 2008) helps us to make the performance tasks (in relation to dealing with ambiguities and long distance relations) tractable.

The last chapter, §4, will provide a full specification of the algorithm/grammar used, discussing which empirical coverage this model can attain; in particular Strong Island conditions (§4.2) will be explained in completely new terms.
Any scientific framework includes a precise definition of the primitive components to be used to describe an empirical reality; these primitives are assumptions the model must postulate in order to be, first, descriptive of a finite set of empirical phenomena under analysis, then, predictive in terms of generalizations on novel (acceptable) data. Coherently with this perspective, the main goal of any generative linguistic theory is to define the primitives that allow us to describe the nature of the linguistic expressions.

This chapter is intended to provide a general introduction to these linguistic entities from a cognitive perspective, that is, by describing them in a way that is coherent with the object description in other cognitive domains, like vision. The aim is to justify the primitives that will be used in this book in a precise, non-redundant and coherent (at least in cognitive terms) way.

The first primitive I will discuss in this chapter is the entity bearing the smallest possible quantity of information: the feature, namely the minimal expression of a property of a linguistic entity. In order to clarify the notion of linguistic entity and its associated primitives, in §1.1 I will explore how features can be related to the idea of information compression and to the generalization we are able to make on novel patterns. Then typologies (§1.1.1, §1.1.4), sources of information (§1.1.2), representations (§1.1.3) and the coexistence of features on the same (lexical) object will be evaluated, eventually ending up with an abstract definition of what a feature structure is (§1.1.5) and how much variation is allowed within this structure across languages (§1.1.6).

The second part of this chapter (§1.2) will investigate how feature structures can combine in order to build bigger meaningful units (§1.2.1), which operations are
allowed (§1.2.1, §1.2.2) and in which domain they properly apply. The computation described in these pages will have a strong derivational flavor, that is, essentially, each relation established among elements will be relevant only at a precise stage of the computation, neither before, nor after. Such restrictive perspective is however able to code many relevant relations, such structural asymmetric relations and non-local dependencies (§1.3).

In the final part of this chapter, I will discuss how the operations on feature structures proposed in §1.2 turn out to be extremely “complex” if they are not “bounded”, where complexity and bounding have to be expressed in terms of memory load and combinatorial possibilities the computation has to consider in order to store/retrieve/elaborate (linguistic) information; in the last paragraph I will tackle this issue, showing how we could significantly reduce the complexity of some problems by chunking the computation in different “phases” (§1.4).

1.1 What's a Feature

To understand why the notion of feature is related to information compression, let us start with a simple example. Suppose we have a retina with 100 input units (a 10x10 matrix) to encode distal stimuli, then only 10 channels to switch the signal received from the retina toward other processing units as shown in (5):

\[
(5) \text{retinal units (10x10 matrix) encoded by a 10 channel hub}
\]
Assume that the input units (the retina) can behave analogically with respect to any distal stimulation (e.g. that is, roughly, the numbers of photons received by the unit), while the 10 channels are only capable of a digital modulation (namely these switches forward a fixed amount of signal only if a certain amount of photons are received). Therefore, we would have ten bits of information to codify an infinite number of potential analogical inputs.

Note that for a wide range of stimuli, accurately perceived by the input units in their diversity, the transmitters will behave in the very same way, namely, they will lose information by flattening the variety received in input to only $2^{10}$ classes of answers (that could be potentially retransmitted to the inner processing units).

This structure could seem pretty inefficient, but answering the same way to different stimuli could be advantageous in terms of the complexity of the decision to make at further stages of processing.

Imagine, for instance, another analogical retina, composed by a 100X100 matrix of units, perfectly interfaced with an “infrared filter”, that is, a 10X10 matrix capable of bringing only one bit of information per pixel (every 10X10 pixels block of the retina is linked to a different unique pixel of the filter). Finely tuning the sensitivity of the filter, we could classify every group of pixels as potentially dangerous (too hot for human contact) or safe for touching. Such a radical classification will avoid a lot of painful contacts, without requiring any further elaboration of the information received from the filter. Then, crucially, our “conscious” behavior could have access only to the 2-colors-filtered information in order to act properly, losing any previous feature that leaded to the 1-bit-color-image:
(6) analogical retina (100x100 matrix) encoded by a 10x10 matrix (1 bit per pixel)

This is an extreme simplification of the kind of categorization that the process of digitalization has to deal with. This is related to the notion of feature, because any pixel of the filter behaves, in fact, as a feature detector expressing a specific (binary) value. In this sense features are the minimal informative units detectable at a specific level of computation that can be elaborated, that means re-transmitted as meaningful inputs to the inner processing units (i.e. further stages of computation).

From a computational point of view, information compression is useful because it reduces the processing complexity both in terms of memory load (having 100 units to “control” is better than having 10,000) and in terms of combinatorial possibilities (patterns on a bigger matrix are more numerous than on a smaller one), biasing the representation of the information on the basis of only a relevant subset of information detected in the input (features). From this perspective, once used by the immediately subsequent stage of the computation, this information becomes inaccessible to further inner elaboration units.

Obviously this outlook of a computational model is neither neutral nor innocent at all: this modular (Fodor 1983), fairly symbolic (Newell & Simon 1972) and hierarchical (Marr 1982) view has been intensively challenged since mid-eighties (e.g. Rumelhart, Hinton & McClelland 1986, Manning & Schütze 1999). From an epistemological point of view it is
We should notice that, in order to be computationally interesting in terms of complexity, features have to be finite in number and discrete in value. If this is not the case we would face a different set of problems: an infinite number of features would prevent the system from being finite (this is because an infinite number of axioms should be specified) or predictive (if we do not specify how the system should behave in presence of certain features, its behavior would be unpredictable).

1.1.1 Features in linguistics

In linguistics (but also in many other fields of cognitive science such as vision studies) features express properties that are usually associated to categorization (and sub-categorization) in morphosyntax, semantics and phonology: the process of identifying classes seems a prerequisite for triggering elaborations leading to complex objects building, quasi-deterministically (that is, extremely fast). These objects are often poorer of accessible (“conscious”) properties (for instance we perceive a 3D object, not the two-dimensional edges fragments that contributed to the 3D mental image, Marr 1982). For these reason, despite his privileged status of “primitive” unit, the notion of feature and the actual number of features needed might be hard to be defined because of their “encapsulated” nature.

Nevertheless, most of the current generative frameworks express crucial properties of the linguistic objects by using features: features determine the position that an object can occupy by triggering its lexical insertion (a transitive verb selects an argument, (7).a Vs. (7).a’)) and/or displacement (wh- elements, in English, stand however much safer starting with these assumptions that are much more restrictive and explanatory.

4 Notice that uncertainty is not necessarily a bad thing: uncertainty, under certain circumstances, is an explicit theoretical assumption in many frameworks such as quantum physics (Pensore 1989).
usually at the beginning of the sentence rather than right after the verb that requires them as arguments, (7).b); features provide essential instructions for the performance systems, namely, the motor-perceptive system and the intentional-conceptual system ([Mary]$^5$) bears phonetic features that can be pronounced and heard, and it has semantic features that allow us to interpret this word as the proper name of an [animate] [human], (7).c$^6$.

(7) a. John [transitive_verb kissed [argument Mary]]
   a'. *John [transitive_verb kissed [determiner the]]
   b. [question Who] did John [kiss [argument _ who]]
   c. Mary = {phon = /m æ r i/; sem = [[animate], [human]...]}  

This is the case of the Government and Binding approach (§2.2.1) and the Minimalist Program (§2.2.2): linguistic objects are considered feature sets composed by phonetic, semantic and formal (selectional, categorial, case etc.) features (Chomsky 1995:21-22). This common assumption brings up a first, naïf, but also logical, question, that is why we should explore the possibility of having (natural) classes of features.

From a theoretical point of view, using classes would avoid the problem of having an unpredictable interaction among linguistic properties: this would lead to an enormous computational load, predicting, moreover, possible grammars in a too permissive way. Identifying macro-classes (or typologies) of features represents an

---

$^5$ From now on, I will include under squared brackets features clusters: [feature_a feature_b ... lexical_item] (unless specified, I will assume that feature_a, feature_b ... are ordered sets).

$^6$ Clear cut categorizations, broadly speaking phonetic/phonological, morpho-syntactic, semantic, etc. are obviously problematic from many point of view; the discussion here goes beyond this simple partition: independently on their status, features will constraint/trigger computation according to their structural configuration.
elegant solution to reduce the potential interaction among properties, distinguishing levels of productive combination and predicting inert coexistences (§1.1.3). There are however reasons to believe that coarse-grained typologies can be further finely explored: for instance, much work in phonology (Clements 1985, Halle et al. 2000) led to a more natural and heuristic hierarchy of features (then properties suitable for hierarchies such as sisterhood, inheritance, motherhood etc. can be defined among features) instead of flat sets (§1.1.5). This idea inspired also some morphosyntactic feature analysis (Harley and Ritter 2002).

The bridge between feature typologies and feature hierarchies will be explored by investigating how and when features are introduced in the (linguistic) computation: in §1.1.4 I will briefly review Chomsky’s idea of inclusiveness (any feature entering the computation should be projected from the lexicon, Chomsky 1995:231) and the distinction between intrinsic features, explicitly valued within the lexical representation of the linguistic object (for example categorial features in regularly suffixed adverbials), and optional features, which are underspecified within the lexical representation of the object (like case in English nouns, or number/gender in the English determiner “the”). This difference aims at distinguishing between feature values that are fully listed in the lexical entries (thus being a proper part of the knowledge representation of the cognitive object) and feature values that are compatible with the lexical entry but not directly specified in it (that is, they are not present in the mental representation of this object taken “in isolation”, but associated to it during the computation).

The last three paragraphs will deal with three other relevant theoretical issues: the equivalence between features and categories within a set-theoretic framework (§1.1.5), the specification of some relevant patterns of distribution (§1.1.6). Finally, some theoretical ranges of parameterization/variation on feature structures will be discussed (§1.1.7).
1.1.2 Feature typologies

Following the standard convention, a lexical entry, that is the smallest meaningful informational cluster within our linguistic knowledge representation, should bear essentially three kinds of features: phonetic, semantic and formal.

This tripartite typology is justified by a modular view of the linguistic cognitive component: different principles/properties may hold at different levels of processing. Roughly speaking, the first two classes include features that are instructions for the performance systems, namely the motor-perceptive system (phonetic features) and the intentional-conceptual one (semantic ones), while the formal features are related to the morpho-syntactic processing unit. This distinction is intuitively plausible since, for instance, phonetic features seem to have nothing to do in terms of structural combination with semantic ones: features spreading or vowel harmony (Clements 1985) are productive phenomena in phonological terms, but they are completely irrelevant for the meaning; the same is true for mass/countable or animate/inanimate distinction: they are intrinsic semantic relations, with related effects on the phrase structure, but with no specific effects on the phonological envelope of the words in most of the natural languages. An easy way to deal with these phenomena would be to assume independent phonetic, semantic and maybe formal (syntactic) representations related to each other by means of correspondence rules (cf. Jackendoff 1997).

From the narrowest perspective (namely the Minimalist Program, §2.2.2), the performance systems are external to the core linguistic (that is syntactic) processing module. This implies that these performance systems can respond to independent generative procedures and to different feature organizations. More precisely, the phonological component interprets a sequence of sensory-motor instructions that have to be provided by phonetic features (e.g. [coronal], [voiced] etc.); the (structured) set of feature reaching the phonological module is called Phonetic Form.
(or simply PF). Note that PF is characterized by a strict linearization requirement imposed on these features\(^7\). This property is often considered insignificant at other levels of processing (surely at the semantic level but, maybe, also at the syntactic one).

On the other hand, the semantic component, which receives in input a Logical Form (for short, LF), operates on conceptual primitives (i.e. [singular], [definite], [referential], [past], [epistemic modality] etc.) that directly affect the meaning of the expression. Among these features, no linear order is needed for interpretive purposes: by the principle of compositionality (Montague 1974) the meaning is largely determined by the hierarchical configuration of conceptual-intentional features. A plausible hypothesis\(^8\) is that only dominance relations should be defined among semantic features.

Formal features represent the third (blurry) typological class, which groups categorial, case and selectional features, among others. The ambiguity of this class derives from the fact that most of these features can hardly have an independent nature with respect to the semantic and the phonological component: this is clear for number and person ϕ-features on nominal phrases, but even case features seem to have semantic correlates (at least they productively restrict the ability of any object bearing case markers to receive any semantic interpretation\(^9\)); on the other hand,

\(^7\) Cf. Fox & Pesetsky 2004. We could predict a certain degree of specificity on this point, related to the phono-acoustic modality; the story could be somehow different for other modalities (cf. Sign Languages).

\(^8\) See chapter 3 for a (formal) discussion on this point.

\(^9\) This is surely true for inherent case (that is dependent from the thematic role assignment); structural case (that is, conversely, structurally determined) would require a more detailed discussion.
selectional features are restricted by semantic properties (the verb “die” takes as subject an animate object), while focus features have evident suprasegmental (i.e. phonological) and semantic correlates. Moreover, dominance relations, but maybe not linear order, seem to play an important role in the structural description of the formal feature organization of a well-formed sentence.

From a wider cognitive perspective, both linearization and dominance relations are relevant in many other processes such as vision, motor control, tactile perception etc. In vision, for instance, we can think of linearization as the topological distribution of features in a perceived scene (8).b, while the dominance relations express a hierarchy of constituency, where dominated elements are parts of the dominating ones (8).c:

(8) a. original image

b. relevant topographic relations

c. constituent structure

```
face
  /   \
eyes  nose  cheekbones
   /     /     |
  pupils
```
Without pushing too far the parallelism between language and vision\(^\text{10}\), we could notice that while phonetic features are clearly distinct from the topographical cues in a scene (points, edges, etc.), even if similar relations seem to hold in both domains, semantic feature structures could be fairly similar to the semantic representation of concepts used in language\(^\text{11}\). Moreover “formal” features hardly find a clear correlate in vision (color properties? movement?). On the other hand, one property, quite accepted in vision science but mostly ignored in generative approaches to language, is the hierarchical structure of the processing (Marr 1982). On the perceptual side\(^\text{12}\), vision starts from edges, bars, virtual lines and blobs detection (primal sketch), which, at a higher level, are combined using fading/shading information in order to lead to a description of the scene in terms shapes/surfaces orientation (2½ dimensional representation). Then the process leads to a complete 3-dimensional description of the perceived objects in terms of surfaces and volumetric primitives hierarchically arranged.

A similarity between Marr’s and Chomsky’s model is that both incorporate (implicitly or explicitly) a “safety device” that prevents the derivation from crashing: features do not enter levels of processing where they are not interpretable: uninterpretable \(\varphi\)-features (like person and number) are deleted from the verb before

\(^{10}\) That clearly are independent cognitive processes even if some underlying properties are significantly similar.

\(^{11}\) Obviously the ontological status of this remark should be deeply evaluated, but this is out of the possibilities of this book.

\(^{12}\) We could imagine a “production” side of this process in terms of abilities of drawing/sculpting “grammatical” pictures/statues (Boccioni’s ungrammaticality would stand to “grammatical sculptures” as Marinetti’s sonnets stand to “grammatical sentences”).
reaching LF in Chomsky’s model\textsuperscript{13}, while bars, virtual lines and blobs do not directly enter the description of 3D objects in Marr’s one.
In sum the typologies of features (such as semantic, phonetic, formal) are justified mainly with respect to the levels of processing at which they can be accessed and elaborated in a principled way (allegedly different from level to level).

1.1.3 Features from the lexicon: the inclusiveness condition
In the previous paragraph, I referred several times to the notion of lexical knowledge without explicitly considering how this information could be represented. Following standard conventions, the lexicon can be thought of as a list of elements bearing the specification of idiosyncratic properties (that is, they are unpredictable on the basis of general principles of linguistic processing). In fact, we could easily hop between static knowledge and general processing instructions, assuming features to be stored in the lexicon or derived by general principles; most of the time it will turn out to be equivalent, at least for computational purposes, as exemplified in (9):

\begin{align*}
\text{a. } & \left[ \text{Determiner, selects-Noun, the} \right] \quad \text{(lexical entry)} \\
\text{b. } & \left[ \text{Determiner the} \right] + \text{Det. always selects a Noun } \quad \text{(lexical entry + principle)}
\end{align*}

A theory that assumes both the lexical entry in (9).a and the principle in (9).b at the same time will be redundant. We would avoid redundancy as far as we do not have empirical evidence of its utility\textsuperscript{14}. A subtle difference between the two formalisms however exists and it is related to the fact that (9).a does not allow for any generalization (namely, it is a language specific lexical entry), while (9).b has a universal flavor and produces substantial limitations of the grammatical generative power: in this sense (9).b seems the correct way of formalizing universal principles.

\textsuperscript{13} This happens essentially by checking, that is, by pairing uninterpretable features on the verb with their interpretable counterpart on the arguments. See §2.2.2 for more details.

\textsuperscript{14} See learnability issues explored in Jackendoff 1997.
Going back to the lexical structure, in §1.1.1 it was explicitly assumed that the lexicon is the place where features are stored and, in addition, it has been suggested that it could be organized on multiple levels (phonetic, semantic and formal). Following these considerations, this could be a representation of the actual shape of a lexical entry:

(10) multiple levels of information in the lexical item “dog”:

Within this perspective, the lexicon is a set of lexical elements \( \text{Lex} \) of the following form:

(11) \( \text{Lex} = \{S, P, F\} \) such that \( S, P \) and \( F \) are respectively finite sets of semantic, phonetic and formal features.

Therefore, this seems to be the locus of the (arbitrary) mapping among levels. To make a little step forward we could investigate the nature of the relationships among these levels of representation, first asking which features are introduced in the derivation and when it happens.

Chomsky tries to answer this question by introducing a condition of inclusiveness (to be thoroughly discussed in §2.2.2):
(12) **Inclusiveness condition** (Chomsky 1995:228):

any structure formed by the computation is constituted of elements\(^{15}\) already present in the lexical items selected for *Numeration*.

*Numeration* is a one-time selection of items from the lexicon that will be available for the computation. For the time being, we are not interested in why this should be a one-time-for-all operation, but just to what extent it makes sense to think of the lexicon, structured as in (10), as the repository of the whole feature information needed to understand/produce a sentence. From this perspective, selecting a lexical element would imply selecting the whole set of features associated to this element in the lexicon. From a performance perspective, however, it is interesting to notice that the notion of numeration makes sense in production, but very little in comprehension: it is plausible, to some extent, that in order to produce a sentence the system has to select a set of lexical items with their complete featural make-up before starting to spell out words; but when we perceive a sentence, we parse it (that is, we assign it a structure) word by word, as soon as speech is perceived (Bever 1970). This operation requires, first, that the lexicon be accessed more than once to retrieve items; second, some retrieved element could be “incomplete” or even wrong in their featural make-up, as shown by garden path effects\(^{16}\):

(13)a. The horse raced past the barn fell   (Bever 1970)

   b. The horse (that was) raced past the barn fell (down)

In (13) a the word “fell” produces a breakdown in parsing, since the reduced relative, disclosed in (13).b, is not recognized at the first reading. This (structural) ambiguity shows that accessing the lexicon and retrieving the relevant (complete) set of features is not a trivial task. Then it seems plausible to assume that we drop

\(^{15}\) Namely features.

\(^{16}\) In §2.2.3 more details will be provided on *Garden Path* and on *Incremental Parsing*. 
unused features (or, better, we do not select them at all) according to a structural hypothesis that we are forced to make as soon as possible, integrating elements piecemeal in the sentence without waiting or requiring a complete numeration. Something similar seems to happen also in vision, since we “interpret” pictures (that is, we assign constituency relations to the perceived elements) even if only a partial view is provided (14).a:

(14)a. input picture

b. interpreted/perceived objects (constituency relations based on detected edges)

c. real objects

Sometimes, other “perspectives” (14).c force us to backtrack from early hypotheses (14).b. From a speculative point of view, we could extend the same considerations to production as well: errors both in language (e.g. slips of tongue, false starts) and in vision (e.g. mis-drawing of lines, difficulty to complete a picture), suggest that even if the “semantic” content of the message is clear to the speaker/draughtsman, its practical realization could be less straightforward. This could be related to the fact
that different classes of features are not always unambiguously associated to a given lexical item, hence they are not directly retrievable any time, at any level of the processing and, crucially, independently from the performance task.

If we are interested in performance task and we want to keep into account parsing and generation, a blocked list of features stored in the lexicon is un-explanatory; an alternative hypothesis, that will be explored in this book, is that feature retrieval is indeed part of our competence and that, depending on the performance perspective (production or comprehension), we access different classes of features trying to integrate dynamically a complete set of features associated to the (lexical) element that we need to parse/produce.

This hypothesis seems to be tenable at least within some generative linguistic frameworks such as Distributed Morphology (Halle and Marantz 1993): in this approach, lexical items are bunches of phonetic features inserted only at PF; before that level, the linguistic computation works only with de-lexicalized sets of semantic-formal features.

This asymmetric computation could suggest a distinction between intrinsic features that are associated to the elements directly in the lexicon (off-line features assignment) and optional features that are associated to these elements during the computation (on-line features assignment, as discussed in Chomsky 1995:277): for instance, the word “we” in English would bear the intrinsic formal features \([\text{first person}], [\text{plural}], [\text{nominative}]\) and the categorial status \([\text{pronominal}]\). These features are present even if the word is taken in isolation; this shows that the lexicon (or the “word shape” in Lexical Functional Grammar terms, Bresnan 2001) provides enough information for off-line features assignment to this item. Words like “John”, on the other hand, can be produced both with a nominative and with an accusative case associated to it, for instance. This is because in our lexicon certain items are underspecified for case features and can receive, dynamically, from other lexical
items (e.g. verbs and prepositions), in specific structural settings, a larger selection of cases than words like “me”.

From a parsing perspective, a similar consideration has to be formulated: consider words like “dog” (“you should dog her” → [verb] Vs. “the dog chases the cat” → [noun]). This is a trivial case of lexical ambiguity (or features underspecification). In these cases, we have three logical possibilities:

a. both the categorial features [verb] and [noun] are selected, then one of them is dropped off during the computation;

b. the selection is procrastinated up to the point where we have more contextual information to identify the unique relevant feature;

c. only one feature is selected every time we try to parse a sentence. If our analysis crashes then we backtrack and consider alternative features.

The last option is more attractive in terms of computational complexity (introducing extra features or delaying available choices, could make the processing time and the memory load grow pretty quickly). This problem is also known as the “multiple tagging problem”: it has been calculated (Derose 1988) that 40% of the word tokens in the Brown Corpus is ambiguous, in terms of syntactic category, even if many of them are very easy to disambiguate: this is because of the different likelihood of the different possible tags. We could describe the problem in terms of likelihood from two different points of view. First, there is a general likelihood for each item in the lexicon to receive specific features (e.g. dog is more likely to be a [noun] than a [verb]): this would be a case of off-line feature assignment (namely this information should be stored somewhere in our lexical entry). Second, the likelihood of a feature is also dependent from other contextual features (what precedes/follows/dominates/is dominated by the element we would assign a tag to raises or lowers the chance of assigning a specific feature to this element). This seems to be a matter of on-line feature assignment, determined by the phrase structure model and probably
deducible from an adequate knowledge structure: for example, in a classical Context-Free Grammatical framework (e.g. Chomsky 1957), the feature for “dog” could be easily predictable following the only plausible top-down expectation as shown in (11)\(^7\):

\[
(15)\text{grammar:}\quad S \rightarrow \text{DP VP}; \text{DP} \rightarrow \text{D N}; \text{VP} \rightarrow \text{V DP}; \text{D} \rightarrow \text{the}; \\
N \rightarrow \text{dog}; N \rightarrow \text{cat}; V \rightarrow \text{dog}; V \rightarrow \text{chases}.
\]

sentence: the dog chases the cat
tagging: the $\leftrightarrow$ D (by unambiguous bottom-up tagging)

\[
\text{dog} \leftrightarrow \text{N/*V} \quad \text{(by top-down expectation: DP > D N)}
\]

chases $\leftrightarrow$ V (by unambiguous bottom-up tagging)

the $\leftrightarrow$ D (by unambiguous bottom-up tagging)
cat $\leftrightarrow$ N (by unambiguous bottom-up tagging)

Summarizing the points discussed in this paragraph:

i. lexical entries should be lists of features that are unpredictable from general principles;

ii. the lexicon could be organized on multiple, structurally independent, layers (crucially semantic, phonetic and, roughly speaking, formal) that could be accessed at different levels of processing;

iii. features are introduced in the derivation from the lexicon or assigned during the computation on the basis of the feature structure, in accordance to specific (probably level dependent) principles of combination;

\(^7\) Upper symbols are non-terminal items (“S”, “NP”…), “S” is the root node (Sentence), lower symbols express terminal items (“cat”, “the”…), “$\rightarrow$” is the rewriting symbol, “$\leftrightarrow$” expresses Part-of-Speech tag assignment. “Bottom-up” indicates an exploration of the rewriting rules from right to left, “top-down” indicates the opposite.
iv. there could be (a)symmetries in the way features enter the computation depending on the performance tasks that access them (comprehension Vs. production).

1.1.4 Feature hierarchies
Here I want to discuss the idea of having hierarchical feature representations as alternative to unstructured sets of properties. For instance, in phonology, a multi-tier hierarchical representation, as in (16), correctly characterizes sets of features that behave coherently with respect to spreading or reduction phenomena, for example, without affecting irrelevant features in different tiers (Clements 1985).

(16) *feature geometry* excerpt in phonology (Clements 1985)

Articulatory parameters (namely classes of features) sometimes present high degrees of independence (laryngeal configuration with respect to opening or closing the nasal cavity) while sometimes they are strictly constrained (spread glottal configuration determines non-voicing). Clements suggests that a hierarchical structure can be predictive of these phenomena: higher-branching categories tend to be more independent than lower ones (Clements 1985:230).
From a formal point of view, we should notice that this model expresses in a very
elegant and compact way two important properties:

i. the complementary distribution of features (for instance \*{\{spread glottal\},
    \{voicing\}}^{18});

ii. the relevant domain of application of specific principles (assimilation works
    well only with laryngeal features);

Even from a syntactic perspective, some empirical data go in this direction: the
cartographic frameworks (§2.2.4) for instance assume that the functional layers^{19},
classically identified as Determiner Phrase^{20}, Inflectional Phrase^{21} and
Complementizer Phrase^{22}, have been, during the last two decades, deeply explored
and split up in subclasses to better express asymmetries in elements distribution.

One of the best examples to explain this idea is Cinque’s (1999) analysis of
adverbials: Cinque suggests that the IP shell is more structured than classically
assumed and that this is a universal generalization supported by robust empirical
evidence. Also in languages that do not have specific morphemes that realize
Cinque’s postulated functional heads, their existence is justified by the distribution
of adverbial elements:

---

18 Unordered sets will be included under curly brackets: \{a, b, c \ldots\}

19 Intend functional to be opposed to lexical: a lexical element bear argumental/eventive
content into the sentence (nouns and verbs, for instance are lexical items); functional items
(such as prepositions, complementizers, determiners etc.) modifies lexical items helping
defining the constituency structure of the phrases, without affecting the lexical content.

20 DP, e.g.: \[DP [D the] [N dog] \]

21 IP, e.g.: \[IP [I does] [VP \ldots] \]

22 CP, e.g.: \[CP [C that] [VP \ldots] \]
With a flat intonation, “probably” (in Italian as in English) has to take scope over “often”. The simplest way to express this intuition is to put the relevant features in a hierarchy that can be roughly expressed by these three basic classes:

(18) modal > temporal > aspectual

This gross classification will be refined (following Cinque) in §2.2.4; for the time being, it is enough to highlight another important property that the Cartographic Approach suggests about hierarchies:

iii. feature hierarchies can be predictive of the relative scope that features have with respect to each other (and, maybe as epiphenomenon, on their linear order when linearization is required);

Turning to the semantic domain, here too feature hierarchies turn out to be relevant. Consider for instance the classic syllogism in (19).a and the non-valid form of (19).b:

(19) a. Men are mortal, Socrates is a man, then Socrates is mortal
    b. *Socrates is mortal, Socrates is a man, then men are mortal

This is just an example to point out how properties are inherited along structures: the validity of (19).a follows from the relation of hyponymy between “Socrates” and “man”: “Socrates” is an hyponym of “man”, then it can inherit some properties form “man”. Since mothers assign properties to daughters (19).a but not vice versa (19).b:

iv. we could predict inheritance relations of some features from hierarchies

Moreover, the meaning of a word can be predictive of its syntactic behavior: Levin (1993) tries, with interesting results, to derive classes of argumental selection from the semantics of the verbs (Levin 1993:2):
“Fill” and “pour” are subject to different constraints involving their arguments selection within the verbal phrase: the first verb is in the class that Levin calls *locative*; according to Levin, crucial aspects of the meaning of the verb that determine its belonging to this class are the facts that it involves a change of location of substances/things and that it affects completely this substance/thing during the action. The syntactic specificity of this class is mainly expressed by the prepositional element that selects the locatum argument (substance or thing subject to the change of location), in this case constrained to be introduced by “with”.

“Pour”, on the other hand, is a member of the class called *benefactive verbs*: these verbs mainly express creation (or preparations as in this case, namely a process that can crucially be incremental). They require a double complement construction as the verb “(to) fill”, but in this case the direct object has to be the locatum argument, and the location has to be introduced by a “to”-like locative preposition (onto, into etc.). This is a good example of classes that do not describe hierarchies, but that suggest the plausibility of systematic relations among levels, in these cases between the semantic and the syntactic level:

v. **interface/mapping conditions** - the structure of the classes (maybe hierarchically organized) at one level (for instance syntactic alternations) could be determined by the properties at other levels (i.e. semantic properties)

Summarizing, feature geometry approach could be extremely useful to represent some important aspects of the lexical knowledge. In particular, it allows for generalizations on:
i. distribution of features
ii. domain of application of specific principles
iii. relative scope (order) among features
iv. properties inheritance
v. mapping relations among levels

A unified hierarchy that represents at the same time features of any level is hardly conceivable within the current linguistic framework. We rather need multiple hierarchies and a nearly deterministic way of mapping these structures across levels (cf. Jackendoff 1997). A further theoretical possibility (that however will not be evaluated in these pages) is that features could be grouped in different ways once we refer to comprehension or to production (feature geometry in phonology is highly predictive in production, but maybe it could be less expressive than the standard theory of distinctive features in comprehension).

I will now introduce the formal devices that allow us to describe the proposed structures. The standard way of representing features structures (for instance in GPSG/HP SG, Pollard and Sag 1994) is by using Attribute-Value Matrices (22).a or Direct Graphs (22).b, the two formalisms are equivalent as shown below:

\[
\begin{align*}
\text{(22) Attribute-Value Matrices} & & \text{Direct Graphs} \\
\begin{cases}
\text{Num} = \text{Sing} \\
\text{Gen} = \text{M} \\
\ldots \\
\text{Feature}_n = \text{Value}_n
\end{cases} & & \begin{array}{c}
\text{Num} \\
\text{Sing} \\
\text{Gen} \\
\text{M}
\end{array}
\end{align*}
\]

Attribute-Value Matrices (henceforth AVM) are finite sets of symbols pairs: assuming \( F \) to be the set of feature sorts, and \( V \) the set of all possible values, an \( AVM \) is a function \( F \rightarrow F \times V \). The value of a feature can be, in fact, either a (single)
symbol of $V$ (i.e. $\text{Num}=\text{Sing}$) or another symbol of $F$ ($\text{Agr}=\text{Num,Gen}$). It is theoretically possible (and acceptable from a grammatical point of view) to have more feature types pointing to the same values as in (23).a but not vice versa, as in (23).b:

\[
\begin{cases}
\text{Feature}_1 = \text{Value}_1 \\
\text{Feature}_2 = \text{Value}_1
\end{cases}
\quad \ast \quad
\begin{cases}
\text{Feature}_1 = \text{Value}_1 \\
\text{Feature}_2 = \text{Value}_2
\end{cases}
\]

(24).a is a case of structure sharing (Pollard and Sag 1994): when features share the same value structure, we can use pointers (or indices) to relate the value structure:

\[
\begin{cases}
\text{Feature}_1 \xrightarrow{1} \text{Feature}_2
\end{cases}
\]

The proposed feature geometries at different levels can be represented as in the following AVMs:

\[
\begin{cases}
\text{Root} \quad \text{supralaryngeal} \quad \text{manner} \\
\text{laryngeal} \quad \text{place}
\end{cases}
\]
b. syntax (based on Cinque 1999):

\[
\begin{align*}
\text{Verbal inflection} & : \\
\text{tense} & : \text{act \ldots} \; \text{valutative \ldots} \; \text{probator \ldots} \; \text{epistemic \ldots} \\
\text{aspect} & : \text{irrealis \ldots} \; \text{past \ldots} \; \text{future \ldots} \; \text{habitual \ldots} \; \text{ripetitive \ldots} \; \text{frequentative \ldots} \; \text{completive \ldots}
\end{align*}
\]

c. semantic (based on Levin 1993):

\[
\begin{align*}
\text{Transitivity} & : \text{subj intran} \; \text{causative \ldots} \\
\text{unexpressed obj} & : \text{unspec obj \ldots} \; \text{body - part obj \ldots} \\
\text{conative} & : \ldots
\end{align*}
\]
1.1.5 Categorial features

In a feature-hierarchical approach, the notion of category is superfluous: the information that an element belongs to a category X can be expressed in terms of features, simply assuming that this element has the (categorial) feature $\chi$. This is a trivial observation from a formal point of view (as noted by Chomsky 1995:381 note 7), but it can have theoretical implications for our model: we could not necessitate different technologies in order to deal with features, nodes and categories (or labels, cf. §1.2.1, §2.2.2); operations on features (cf. §1.2) can apply to all these objects (for this reason it is more accurate to refer to what has been historically named “category”, as categorial feature).

Intuitively, the concept of (categorial) feature is equivalent to the mathematical notion of set. In fact, when we refer to a grammatical category, for instance “noun”, we refer to a homogeneous domain in which the included elements share some configurational, semantic or phonological properties. We usually draw trees for representing categorial inclusion but, in fact, the representations in (26) are equivalent:

\[
(26) \quad \begin{array}{ccc}
\text{a. } & A & \text{b. } \\
& a & \text{c. } A[[a][b]] \\
& b & \\
\end{array}
\]

In fact, we consider the words “cat”, “chase”, “white” respectively as a noun, a verb and an adjective, because “cat” behaves the same, at least from a syntactic point of view, as words like “dog”, “man” or “mouse”, while “chase” behaves as “beat” or “push” and “white” as “sick” or “fat”. Roughly speaking, when two elements can be substituted in a structure without affecting the grammaticality of the sentence, they belong in the same category:
(27) a. The cat chases the white mouse
   b. The dog chases the big man
   c. The cat beats the fat mouse
   d. *The push chase the big mouse
   e. *The cat white the fat mouse
   f. *The cat chases the drink mouse

Computationally, this behavior has been classically caught, for example, by rewriting rules in Context-Free Grammars:

(28) Determiner Phrase → D N;

\[
\begin{align*}
D & \rightarrow \text{the}; & D & \rightarrow \text{a}; & D & \rightarrow \text{this}; \\
N & \rightarrow \text{book}; & N & \rightarrow \text{boy}; & N & \rightarrow \text{dog};
\end{align*}
\]

The relation between the categorial feature D, or N, and the lexical entries is a one-to-many relation; this means that if the categorization is productive, it allows us to generalize a syntactic behavior, describing it in a compact way as shown in (28).

The advantages seem neat in terms of expressive power, but there are at least two tricky questions to be answered:

- What allows us to associate a feature to an element?
- How many features do we need in order to represent the lexicon?

Answering these questions is, again, mainly an empirical matter and looking at performance tasks could suggest us some interesting strategy: to attempt an answer to the first question, let us concentrate on comprehension. We should observe that language, even if theoretically pervaded by ambiguities, is a rather efficient medium for information transmission; ambiguities are most of the time readily solved by the human interpreter (in fact, most of the time, the interpreter does not even perceive
the “theoretical ambiguities” that artificial parsers can find\(^{23}\). But what the human hearer captures as input is just a signal. The shape of this signal and the order of its subparts are the only possible clues for feature retrieval. This is the outline of the problem at any level of processing: from a phonetic point of view, features\(^{24}\) are detectable from relative changes in wave formants; this analysis requires accessing the “shape” of the signal and interpreting it in an ordered (temporal) way to appreciate productive differences. From a syntactic point of view, as LFG overtly assumes (Bresnan 2001), the dominance/scope relations among elements can be predicted either by the element shape (for instance, case markers are usually assumed to be the main source of structural information in non-configurational languages such as Warlpiri, but see Legate 2002) or by the relative word position (as in English). From a semantic perspective, the word shape is a crucial link to the meaning, but the latter is often primed by the context (in a broad sense, what has already been presented/structured before the word we are analyzing). What we can say at this point is that, at any level, first the shape of the element then the expectations produced by the context help us in finding the (under)specified features to be associated to the element.

As for the second question (how many features the lexicon could be represented with?), it is easy to guess that the answer should be something like “as few as possible, avoiding overgeneralizations”. No more than this could be said from a

\(^{23}\) “I saw the man in the park with a binoculars” has three parses, only the first one is most of the time considered by humans:
a. I’m watching with a binoculars
b. I’m watching a man that has a binoculars
c. I’m watching a man in the park where there is a binocular.

\(^{24}\) Assume distinctive features standard theory.
theoretical point of view. Assuming anyway a finite number of (categorial) features, a related problem is the nature of the sets generated by the choice of these features: many years ago, Ross addressed a similar issue: the “category squish” (Ross 1972). The empirical observation was that Nouns, Adjectives, Participials and Verbs constituted gross classes in which elements sometimes have a border-line behavior (with respect to their relative order, or to some specific structural requirement); this makes it difficult to assign these elements to a unique category. Ross then suggested a continuum of the classical categories (N, Adj, Part, V) on which elements could be scattered. From a pure formal point of view, our framework should not allow for fuzzy/blurred categories: features are discrete entities; an element can bear one feature or not, namely, it can belong or not to a class without other possibilities. Nonetheless, the phenomena discussed by Ross can be easily explained in formal terms. Suppose we have six elements \{a, b, c, d, e\} with associated the set of features \{\alpha, \beta, \gamma\} in the following way:

\begin{align*}
(a \rightarrow \{\alpha\}; b \rightarrow \{\alpha, \beta\}; c \rightarrow \{\beta\}; d \rightarrow \{\beta, \gamma\}; e \rightarrow \{\gamma\};
\end{align*}

Every feature creates a set, for simplicity let us say that elements with feature \alpha are Nouns, elements with feature \beta are Adjectives and elements with feature \gamma are Verbs. The graphic representation of these data is expressed below:

\[(30)\]

![Diagram](diagram.png)
The elements $b$ and $d$ would be ambiguous if we should assign them a unique category; from this perspective, sets are clear-cut entities and the ambiguous behavior of some elements is due to the co-presence, within the same linguistic entity, of single features (or pattern of features) whose presence, per se, triggers a specific categorial status. Concretely, this suggests that the traditional “grammatical categories” might be epiphenomena determined by the interaction of more fine-grained features (as suggested by “nanosyntactic” research, Starke 2009). We could prevent that from happening in many ways, for example refining the sets (by a subcategorization operation) in the following way:

(31) \{a\} → Nouns; \{a, β\} → Special_Adjective; \{β\} → Adjective ...

From a cross-linguistic perspective, we could observe, that features are assigned to lexical elements in a very diverse way across languages (at least as far as morphology and syntax are concerned). In this direction, the scattering principle proposed by Giorgi and Pianesi (Giorgi and Pianesi 1997) proposes that a bunch of features can be syncretically projected on the same head (that is, the same lexical item) or scattered on various (functional) heads as far as we get unique assignments: \{functional_feature\} → (functional)head. Accepting a similar point of view, it would be possible to relegate this problem to a matter of parameterization. This seems to be a fairly suitable solution by now (Cf. §1.1.7).

1.1.6 Features structures: x-bar theory and functional sequences

If we look at the pictures in (32) we should notice that even if there are features that can be moved or modified, changing productively the “semantic content” of the picture, these modifications can happen only according to some precise rules:

(32) a. b. c.* d.* e.*
While a face with its components (eyes, mouth, hair... and expression) is easily recognized in (32).a-b, this is much harder when:

- features are randomly arranged (32).c;
- features do not agree (in (32).d, for instance, colors and edges do not agree in position);
- something required is missing (32).e.

In this respect, language is quite alike vision and these constraints are incorporated in many generative frameworks. An important constraint on feature structures is represented by the *X-bar theory* (Chomsky 1970, §2.2.1). This is the schema representing the generalizations on phrase structure to be captured:

\[
(33) \quad \text{XP} \\
\quad \text{YP} \quad \text{X'} \quad \text{ZP} \\
\quad \quad \text{specifier} \quad \text{complement} \\
\quad \quad \text{head} \\
\]

Any well-formed phrase seems to conform to this pattern, where the *head* is the hub of the structure (a noun, a verb etc.), the *complement(s)* is the required element the head selects (e.g. any transitive verb selects a direct object) and the *specifier* expresses some further properties of the head, virtually optional unless specified (e.g. before Abney’s *DP hypothesis*, Abney 1987, a determiner, like an article, specified the definiteness of the noun phrase). Any head seems to project a structure like the one expressed in (33) and some distributional constraints can be clearly stated (note the parallel with the constraints expressed in (32)):

- the specifier is mostly (or always depending on the theory, cf. Kayne 1994) to the left of the head, while the complements position can vary (again, depending on the theory) across languages, even though it is (generally)
stable within the same language (head-final languages, e.g. Japanese, are languages where complements precede their head, while in head-initial languages, e.g. English, it is the head that precedes its complements);

- the specifier agrees (at least in person, number and gender, where these features are specified in the language) with their head;
- complements, if required (that is, selected) by the head, have to be present in the structure. If not the sentence is ungrammatical.

Starke (Starke 2001:157) points out that X-bar theory is at least redundant, if not completely misleading, if we assume at the same time a highly articulated functional hierarchy (like Cinque’s): the combination of these two theories would predict duplicated positions (then features) for every functional element (specifier position plus head position). It is in fact very hard, if they exist at all, to find languages that fill at the same time the specifier and the head position, doubly checking the same feature (doubly-filled nothing, Starke 2002). In this sense, the sequence of functional elements (let us call it $F_{seq}$ following Starke 2001) is an adequate starting point that could replaces X-bar theory; on the other hand $F_{seq}$ is not so advantageous in terms of cross-categorial generalizations: what X-bar theory explicitly predicts and what $F_{seq}$ has to postulate with independent hierarchies, are the similarities among phrase structures. Recently, some similarity has been pushed quite far, so to describe interesting parallel (functional) structures both in nominal and in verbal domain (this would prevent the grammar from postulating independent $F_{seq}$s):

i. **nominalization**, the same morpheme can get both verbal (“destroy”) or nominal (“destruction”) morphology;

ii. **adjectival counterpart of adverbial forms**, a great part of adverbial forms (“quickly”) in the VP domain have equivalent adjectival forms (“quick”) in NP domain;
iii. *ontological similarities*, there are some ontological similarities between nominal and verbal properties (mass/countable and atelic/telic distinction, Verkuyl 1999, tenses and pronouns, Partee 1973, modality and specificity, Filip 2000), moreover these properties seem to interact in some way (Specific Quantity Hypothesis, Verkuyl 1999);

iv. *delays in acquisition*, children show similar deficiencies in early stage of language acquisition in both domains (for example optional infinitives in verbal functional domain, Wexler 1994, bare nouns in nominal functional domain, Hoekstra and Hyams 1995).

Economy considerations suggest dealing with these facts exploiting as much as possible these productive similarities. Ogawa (2001), for instance, suggests that the functional shell above NP and VP is similar in many ways. Even though there are many theoretical advantages in accepting this intuition, this is not a problem-free solution and these problematic aspects cannot be sufficiently addressed within this book. For the time being, I will then just try to describe some general regularities as follows without forcing any functional parallelism:

(34) *structural projection principle*

any lexical element (Lex), either a Noun or a Verb, selected to enter the computation, projects a structural skeleton SK consisting of a finite set of selectional constraints (Csel) and a finite set of functional specifications (Fseq) based on the formal/semantic features F present in Lex:

$$SK(\text{Lex}\{F\}) \rightarrow Fseq + Csel$$

This is a graphical representation of the principle:
Notice that this might be an adequate description of phrase structure, but it can not be explanatory, since it does not tell us how comes that phrase structures gets shaped this way.

1.1.7 Features parameterization: how to account for variation

The last issue I want to discuss in this chapter is how features (structures) can vary across languages. It is a common assumption that any natural language selects a subset of features from a universal set of possibilities (Chomsky 2001:4). This is quite clear for the phonological component: we know that newborns are sensitive to phonetic distinctions that are not present in the mother tongue and that then they “lose” this ability at later ages (Eimas 1975); this assumption is much more controversial for syntactic and semantic features, since there is no evidence, at least

\[ (35) \]

\[ \text{Lex \{F\}} \]

\[ \text{head} \]

\[ C_1 \]

\[ C_{sel} \]

\[ C_2 \]

\[ F_{seq} \]

\[ F_n \]

\[ F_1 \]

---

25 See Kuhl (1994) and related work for a more precise definition of the inaccurate idea of “loss” of discriminative ability.
to my knowledge, showing any loss of discrimination between semantic/syntactic properties (for instance, states Vs events or nouns Vs. verbs) to which infants were sensitive. Definitely, that could be related to the difficulty of testing these properties with newborns, but, provided that Sapir-Worf hypothesis has been discredited (Miller & Johnson-Laird 1976) and translation among languages is hard but possible, we can assume at least that semantic features (and their organization) are pretty much the same across languages (cf. Chomsky 2001:4, but see Chierchia 1998 for an alternative proposal).

Starting from Chomsky 1981, parametric approaches attained a good descriptive power in terms of predicted cross-linguistic variations by using a relative little number of parameters: *head-complement position, pro-drop/non-pro-drop*\(^{26}\), *constituent movements*. The leading idea was to consider parameters as related only to formal features (or, more restrictively, only to formal features of functional elements: Belletti and Rizzi 1988, Fukui 1986, Borer 1984), particularly in the mapping between these features and elements in the lexicon; recent evolutions of this framework accept the assumption that parameterization is restricted to the lexicon (Chomsky 2001:4). More generally, following the classical perspective, parameters look like variables playing a role in the application of universal principles, not as functions that directly select features or modify their structure.

From a formal perspective, this notion of parameterization can be described at least in three ways:

\(^{26}\) This parameter controls the optionality of the subject realization: in *pro-drop* languages the subject can be non-overtly realized (in this sense *pro* is the pronominal null subject, e.g. Italian), while in non-pro-drop languages this option is not available and even if semantically empty (expletives) it has to be present (e.g. English), Jaegli & Safir 1989.
(36) three kind of parameterization on features:

a. **subset parameterization** - a proper Subset of Features (Fₛ) is selected from the Universal Feature domain at some specific level (Fᵤ) by a language specific function (P_subset):

\[ P_{\text{subset}}(F_u) \rightarrow F_S \] (e.g. universal set of features: \{A, B, C\} → language selected features \{A, B\})

b. **structure parameterization** - a language specific function (P_structure) maps Universal Features Structures (FSᵤ) to different ones at the same level of processing (FSₛ):

\[ P_{\text{structure}}(FS_u) \rightarrow FS_S \] (e.g. universal structure of features:

\[[\Lambda [\Lambda [c]]]] \rightarrow \text{language specific structure of features} \[[\Lambda [\Lambda [c]]]]\)

c. **mapping parameterization** - a language specific function maps a set of Features at one level (F₁) to single elements at different levels (eₘ):

\[ P_{\text{mapping}}(F_1) \rightarrow e_m \] (e.g. set of features at some level: \{A, B, C\} → language specific mapping \{[\Lambda, X], [C, Y]\})

(36).a is the simplest case of phonetic parameterization, for instance:

\[ P_{\text{subset, Chinese}}(\{\text{consonant, sonorant, voice, continuous, coronal, anterior, lateral}\}) \rightarrow \{\text{consonant, sonorant, voice, continuous, coronal, anterior}\}. \] In this case, speakers of this hypothetical (Chinese-like) language would not perceive any distinction between /l/ and /r/.

Structural parameterization (36).b is a tricky possibility: if we would have this option and any language would implement it, how could we assume that a universal feature structure exists at all? Many empirical results actually show the opposite: even if from surface phenomena (for example word order) we could guess that features structures vary across-languages (e.g. the functional structure above VP), there are many reasons to believe that these are only apparent counterexamples to a
universal feature hierarchy (Cinque 1999). I will assume that (36).b is just a logical abstract possibility in fact not considered by human languages. On the contrary, (36).c is, in my opinion, the most pervasive way of parameterizing features: the lexicon can be considered as an instantiation of functions like these ones; a lexical item can be seen as a set of functions mapping the “word level” to others levels roughly in the following way:

\[ P_{\text{mapping}}(\langle d/\cdot o/\cdot g/\rangle_{\text{phonology}}) \rightarrow \text{dog}_{\text{lexical root}} \]
\[ P_{\text{mapping}}(\{N\ldots\}_{\text{syntax}}) \rightarrow \text{dog}_{\text{lexical root}} \]
\[ P_{\text{mapping}}(\{\text{animate, countable}\ldots\}_{\text{semantics}}) \rightarrow \text{dog}_{\text{lexical root}} \]

The same is true for principles such as the scattering principle (Giorgi & Pianesi 1997, briefly introduced in §1.1.4), which maps functional features to categorial features (namely to features at a different hierarchical level) in the following way:

\[ P_{\text{mapping}}(\{\text{anterior, terminated}\ldots\}_{\text{syntactic features}}) \rightarrow T_{\text{syntactic feature at higher level}} \]

1.2 Basic Operations on Features

Finding features and classifying them is clearly not the unique task a cognitive process is responsible for: in order to capture the generative capacity of the human cognition, we should assume that novel patterns are not just recognized as bringing specific features, then belonging to specific classes, but also arranged in some structural scaffolding that supports an infinite (even if constrained) number of original, compositionally meaningful, combinations. To explain these facts, in this chapter, I will try to investigate how features can be recursively combined depending on their nature and their interaction. In fact, as I presented them, features seem to be mostly inert entities, which do not specify, per se, how linguistic objects can be (un)built. Even though feature structures potentially represent powerful constraints on many possible operations, if we do not explicit the dynamics that

27 But see Keenan and Stabler 2004.
allows them to be arranged in a precise structural way, these features would be completely useless. We call structure building operations (Chomsky 1995) the procedures that allow us to combine features and predict their dynamics. Following recent generative trends (c.f. §2.2) I will explore two operations, merge and move, attempting a speculative comparison with respect to vision in order to highlight some important cognitive properties:

prevent that from happening in many ways, for example refining the sets (by a subcategorization operation) in the following way:

(37) **Merge**

it is the core (recursive) building operation; when objects are combined (paired) together, this operation determines the feature structure of the resulting object;

(38) **Move**

it is an alternative building operation able to interpret an incomplete set of features (essentially in terms of $\pi$ or $\sigma$) using features of a distant (that is, non-adjacent) object, then re-merging the interpreted element in the required position.

Below I will discuss the standard assumptions that allow us to formalize these operations as proper part of our grammatical knowledge. In the rest of the chapter I will introduce a principled distinction between derivational and representational approaches to the competence description (§1.3). Even if this discussion will be explored in more detail in §2.2, I would address the issue concerning the fact that many geometrical relations definable on a whole tree representation (such as $\text{Fseq}$ generalizations, $\text{C-command}$ relations) can be reformulated in derivational terms without any loss of descriptive adequacy, moreover gaining in cognitive plausibility and in terms of computational complexity reduction (§1.4).
1.2.1 Merge as unification

Before exploring linguistic data, let us start playing with pictures in order to exemplify some potentially interesting facts:

(39)a. 

b. 

c. 

d. i. ii. iii. iv. 

e. i. ii. iii. iv. 

The examples in (39) tell us something about grouping. This “operation” is possible if the objects are “close enough” (like in (39).a but not in (39).b) and, in general, if they share some relevant features (39).a,c; which features can be shared in order to trigger grouping is ultimately an empirical matter: for instance, while the result of the grouping operation in (39).a is pretty neat (the dots are combined in groups of two as result of their “compatible” positional features), the groups in (39).c are much more obscure: color (grey intensity), shape, simple proximity seem to be able to trigger a group status but with different intensities\(^28\). On the other hand, (39).d shows that two different features present in the same object (e.g. position and

\(^{28}\) See Palmer 1999 for a detailed discussion on grouping.
...luminosity) can be used (separately) to create two different groups, namely a circle-group and a square-group. Both objects are visible in (39).d.ii, while decomposing the dots in (39).d.iii and (39).d.iv, the former (39).d.iii looses the ability to look like a clean circle-group as the one in (39).d.i (this shows that the elements, with their positional features, belonging to the square-group are indeed crucial to complete the circle-group even if engaged in another grouping). Finally (39).e shows that groups can be overridden, depending on the context ((39).e.i Vs. (39).e.ii)\(^29\); moreover there are properties in the group (parallelism between the group of dots and a line (39).e.iii) that are not present in their single components (a dot cannot be parallel to a line, (39).e.iv). These apparently unrelated phenomena show, indeed, properties that seem to be present also in language (with some fundamental differences):

(40)a. [[the dog] runs]
   b. '[[the dog [that Mary found in front of the door of her office when she was running out to say bye to Joan that was walking on the street at that time]] runs]
   c. i. [the dog]
      ii. *[the was]
      iii. ?[probably yesterday]
      iv. [probably yesterday [John [saw Mary]]]
   d. *John [a kisses [b [Mary] a] dances b]
   e. i. [John [saw Mary]]
      ii. [[The father of John][saw Mary]] Vs. The father of *[John [saw Mary]]
      iii. *[[dog] runs] *[[the] runs]

\(^29\) See garden path data in §2.2.3 for more details on this point.
The parallel with the data reported in (39) is suggestive: we can say that words can be “grouped” if they are in a “local” structural configuration ((40).a Vs. (40).b) and if they bear compatible features (40).c; moreover groups are not simply defined only on the basis of the properties of their elements but they can be overridden depending on the context ((40).c.iii-iv, (40).e.i-ii). Eventually, groups show properties that do not belong to their single components ((40).a Vs. (40).e.iii). “Groups” seem to be created also in language. In fact, there is a very precise way to build groups with “words”: a principled way to describe the emergence of these structures has been (re)introduced in generative linguistics by the Minimalist Program (Chomsky 1995, §2.2.2) with the Merge idea; this operation is assumed to be the basic structure building component: it basically takes a pair of computationally well-formed objects, A and B, and replaces them by a new object C computationally well-formed as well. As we mentioned before, there are important differences between merge in language and grouping in vision. At least two differences are worth to be mentioned here:

- Merge, in language, is assumed to be a binary function; this very same constraint is not evident for grouping in vision (in (39).d.i dots can enter the grouping relation all at once);
- in vision, but not in language, the very same object can enter multiple grouping relations as shown by the opposition (39).d Vs. (40).d.

Despite these idiosyncrasies, it seems possible to describe some common properties underlined by both processes; from this perspective, (a sort of) Merge (both in vision and in language) happens if and only if:

- a relevant local configuration is met among elements that are combined;
- the feature structures of these elements is compatible.

Turning to the structure of the resulting linguistic object, standard assumptions (Chomsky 1995:243) suggest that this new entity C should have, minimally, the
form $\gamma\{A, B\}$ where $A$ and $B$ are the merged constituents and $\gamma$ is an identifier, called label, that would express the category which $C$ belongs to. Economy conditions lead one to assume that we should restrict as much as possible the searching space when we build new objects, then the optimal solution would be accessing only $A$ and $B$ in order to build $C$. Thinking recursively (i.e. Merge after Merge), after the first Merge operation, $C$ should be (informationally) rich enough to dispense any further operation that applies to it, to retrieve its previous constituents. This is a radical assumption, hardly sustainable within the standard minimalist framework (as discussed by Brody, 2002\textsuperscript{30}), but it seems the natural (null) hypothesis from a computational perspective.

For the time being, to understand what kind of information (namely which feature structure) should compose the object $C$, let us analyze four theoretical possibilities\textsuperscript{31}:

(41) a. $C$ is the intersection of $A$ and $B$;
b. $C$ is the union of $A$ and $B$;
c. $C$ is either $A$ or $B$;

Logical considerations on possible outputs rule out both (41).a (besides being null (42).a, the intersection between $A$ and $B$ could be irrelevant, (42).a') and (41).b (the union between $\alpha$ and $\beta$ could be contradictory (42).b or inconsistent (42).b'):

(42) a. $[\text{Adverbial } A] \cap [\text{Verb } B] = \emptyset$
a'. $[D, \text{specific, sing } A] \cap [N, \text{animate, sing, masc } B] = \gamma[\text{sing } ]$
b. $[D, \text{sing } A] \cup [N, \text{plur, masc } B] = [\gamma(D,N), *\text{(sing, plur), masc } ]$
b'. $[V, \text{transitive } A] \cup [N, \text{dative } B] = [\gamma(V,N), \#\text{(transitive, dative) } ]$

\textsuperscript{30} Any constituent, that is a result of a merge operation, has to be accessible in its subparts in order to account for Movement (cf. §2.2 and §3).

\textsuperscript{31} This extends Chomsky’s discussion, Chomsky 1995:244.
(41).c is the solution provided by Chomsky, but his discussion is driven simply by the necessity to derive the label, rather than the whole set of features of C that percolates to the next object. From the opposition (40).a Vs. (40).e.iii, it is clear that some feature (e.g. DP) is not present in any of the components of the phrase (e.g. neither in [D the] nor in [N dog]), but, generally, in the ordered combination of the elements (i.e. <[D the], [N dog]>); in this sense (41).c, namely the selection of only one element, could not be the whole story and, definitely, it would not prevent further operations from accessing the constituent structure.

A related issue, which probably could help us understanding the featural nature of the merge outcome, is about what triggers the merge operation: the simplest hypothesis to be evaluated is that either A or B has at least one feature that requires merge. This is quite in line with what we usually call selectional requirements (standard C(ategorial)-selection but also S(emantic)-selection, Pesetsky 1982):

(43)

a. *[John [kisses ∅]]
   b. *[John [kisses girl]]
   c. [John [kisses [the girl]]]
   d. *[John [kiss [the girl]]]
   e. [John often [kisses [the girl]]]
   f. [[John [kisses [the girl]]][in the park]]

It is a property of the verb “kiss” to require two arguments, an agent and a patient (this expresses the verb valence, Pollard and Sag 1994), in order to meet grammaticality ((43).a Vs. (43).c). Going back to the phrase structure discussion (§1.1.5) we should notice that this kind of requirement was expressed exactly by the C(omplement)S(election) domain. Then we could conclude that CSel features trigger a first kind of merge. There are reasons (to be explored in §3) to believe that this is not the only kind of merge-trigger. Many examples in (39) cannot be easily accounted for in terms of selectional requirement: grouping seems indeed optional in
many cases, since the absence of part of the elements present in the pictures would not produce “ungrammatical” images. A parallel phenomenon in language is expressed by the “optionality” of elements such as adverbials in (40).c.iii-iv, (40).e or adjunct phrase (43).f. Their presence is compatible with the grouping/merge options, but it cannot be accounted for simply in terms of selection. Then some “compatibility check” is required and it does not inspect only the lexical information, but rather the grammatical knowledge about larger phrase structure possibilities (e.g. F(unctional)Seq(iences)). Note that a similar “compatibility check” should be present also to rule out merging operations such as the one presented in (43).b (in this case the argument does not have the required argumental feature) and (43).d (since functional position are involved, i.e. the subject position, agreement is required as postulated, even though not explained, in §1.1.5). Both sorts of merge have in common the ability to prevent the same operation from being repeated more than once:

\[(43)\]  
\[g. \quad *[\text{John often} \{\text{kisses} \{\text{the girl} \} \{\text{Mary}\}]\]  
\[h. \quad ??[\text{John} \{[\text{frequently}] \{\text{kisses} \{\text{the girl} \}\} \{\text{with frequency}\}]]\]

The verb “kiss” once satisfied its requirements (once saturated, in Pollard and Sag’s 1994 terminology) should not show any other selectional requirement (43).g, then we could simply assume its selectional features are “removed” once satisfied; on the other hand, even when a functional specification has been expressed, there is no more room for other elements bearing the very same feature (43).h. These facts are important cues that suggest us what survives after Merge. An important (computational) option to be seriously evaluated, since (partially) compatible with the previous observations, is that Merge be nothing but a unification algorithm.

\[32\) Where “ungrammatical” means that the pictorial elements did not attained a specific “object status”.

56
(44) unification algorithm (adapted from Shieber 1986)

given two sets of features, A and B, their unification C, expressed by \( C = A \cup B \), is the smallest possible set that comprehends all features present either in A or in B without duplications. The unification is undefined when features are incompatible (either they clash in value (42).b, or they are inconsistent (42).b').

This seems to be a natural refinement of the option given in (41).b, but it would conflict with the (maybe apparent) evidence that, sometimes, new features appear ([\[DP\]] from [[\[D\]][\[N\]]]), sometimes they disappear (for instance selectional requirements, when satisfied, should be removed from the resulting object). Even though the first problem (introduction of new features in the structure) disappear if we accept the Inclusiveness Condition discussed in §1.1.3, the second problem is still present and, in this sense, Chomsky’s theory and the unification algorithm would be somehow incompatible. In the rest of this book, I will try to pursue the idea that this incompatibility is only apparent and that the unification algorithm corresponds to the notion of Merge if we accept few departures from the standard minimalist view, namely:

- CSel are not simple features, but expectations on the phrase structure (then they do not need to be “deleted”, since they simply express a required structural projection to be satisfied “after” the lexical head that have these selectional needs);
- [\[D\]] and [\[N\]] are not incompatible labels (then they are not different values for the same Cat(egorical) feature), but, indeed, [\[N\]] is a lexical head feature

---

33 Jackendoff 1997:13 explicitly states that none of Chomsky’s theories incorporates unification algorithms.

34 This issue will be explored in §3.4.3.
and [D] a functional one and they are compatible, according to the Fseq projected by the lexical nominal head and potentially coexistent in phrase structures like the following ones: [DN [D the ] [D dog ] ] or [DN John].

We can now summarize the relevant points discussed in this paragraph: Merge is the basic (simplest) operation for (recursively) building linguistic objects. It seems plausible to assume that many properties of this operation are in common with other cognitive systems (e.g. vision), essentially:

i. it is an operation on features;
ii. it unifies feature structures (checking their compatibility);
iii. its scope is strictly local.

The simplest hypothesis is that what makes this Merge operation special for the linguistic (Vs. visual) system is just the feature structure it applies to and the nature/number of pathways able to process these features (in parallel).

1.2.2 Non-local dependencies: move and merge again

It is a matter of fact that language can express in a variety of ways non-local dependencies between elements:

(45) a. [X What ] , do you think [Y _ ] ? (movement)
   b. [X John ] gave [Y his ] picture to Mary (pronominal binding)
   c. John [X bought the book] and Mary did too [Y _ ] (ellipsis)

This (non exhaustive) list illustrates a relevant set of phenomena such that an element X determines the “interpretation” of a distal (that is, non-adjacent) pronominal/empty object Y (the indices express this “binding” relation). Before looking at specific linguistic properties that regulates these dependencies, we could note now that something similar can happen in vision too:
An occulted corner (46).a is interpreted as (46).a' rather than (46).a''; this fact could be accounted for in terms of “local coherence” (two straight convergent lines are supposed to joint at some point forming the hidden corner\textsuperscript{35}). It is however plausible to assume that not only local features contribute to the interpretation of a hidden element: (46).a'' can be easily mis-interpreted as (46).a', while this is much more difficult in (46).a' where a single feature (an unambiguous straight corner) disambiguate all other hidden corners (even the opposite, non adjacent, one). This “non-local” relation is maybe more clear in contexts where some sort of “pronominal binding” is realized: the convexity/concavity feature on the corner in (46).b can be disambiguated depending on the “context” (the closure side determines the convexity). This is however a property of the whole figure and not only of the

\textsuperscript{35}This is a principle of \textit{figural simplicity} or Prägnanz, Palmer 1999:288.
local geometry of the sides next to the corner as shown in (46).b″. Finally, simplicity (e.g. symmetry, Palmer 1999) cannot be the whole story, since familiar shapes such as a Fiat 500, can hardly be interpreted as (46).c″ when a part of the picture is occulted (46).c.

The reckless parallelism between vision and language resides on the fact that a hidden part of the object (that is, an empty element or trace) can be interpreted (or reconstructed) depending on the visible features (prounced words) that enter in a relevant structural configuration with the hidden part (as in movement or ellipsis), triggering a clear expectation on the whole picture\(^{36}\) (an empty element in a grammatical sentence). On the other hand, a visible ambiguous part of an object (namely an element with some underspecified features like a pronominal form) can be interpreted differently (that is, these underspecified features can be valued) depending on the structural configuration of the features in the picture (pretty much like pronominal binding). With respect to language, the nature/necessity of a move operation, (45).a, in a grammatical framework that already implements merge has recently been questioned (Kitahara 1994, Chomsky 1995-2001, Epstein et al. 1998, Starke 2001 among others). These discussions are partially justified by the observation that any movement operation, indeed, should require a re-merge of the moved element in another (relatively distant) structural position. Whether or not this “reduction” is a real option, we could look at these phenomena from a wider perspective in order to capture some relevant properties that are not evident at all if we stick to the idea that a “chain” (the sequence of elements related by the movement operation) is a linguistic object per se, rather than an epiphenomenon of some deeper structure building process. This second option is maybe more

\(^{36}\) A difference with respect to the language resides on the fact that an expectation in vision can be more easily deceived without necessarily lead to “ungrammaticality”.
perceivable in vision, where the interpretation of an incomplete element (either hidden or ambiguous with respect to some feature) is triggered essentially by two factors:

- the perception of a missing/incomplete element/feature so to fulfill an “object” status;
- the presence of an active pattern perceived as structurally compatible with the incomplete element.

We mentioned earlier that the perception of a missing/incomplete element within an object triggers the expectation for this part; once this expectation is expressed, the active pattern(s) can be inspected in order to fulfill this requirement. Translating this analysis to language, we could try to interpret movement in (45).a in terms of expectation triggers and active patterns: as we pointed out in the previous paragraph, CSel “features” cause expectations, then if, right after the lexical item bearing these expectations, we do not find anything suitable to be integrated, these expectations should cause the inspection of the active pattern(s). Both in vision and in language, we can think of active patterns as to elements present in a sort of accessible memory (maybe a short-term memory). Since only “structurally compatible” items can be used to fulfill a precise expectation, we can assume that only specific elements should be “moved” in this accessible memory. In (46).a the compatibility with the square hypothesis “activates” the complete corner(s), while in (45).a the compatibility with a full sentence hypothesis activates “what” as a potential argument of a potential verbal phrase (not yet present in the structure). When the verbal head is found and a missing argument expected, the accessible memory can be inspected in order to complete the sentence. This last step requires a re-merge (to be intended as unification of the relevant feature structure) of the active pattern in memory.
Many other points should be discussed in order to make this parallelism concret, but for the time being, this should be enough to get a glimpse of the cognitive insight that justifies the formal analysis of movement that will be provided in chapter 3

1.3 Relevant Relations among Elements: Introduction to Derivations

To define what an “active pattern” is, we could start analyzing which configurations among elements are required in order to create triggering contexts (for instance, for movement to happen). First, it is trivial to notice that the relevant relationships we can define among linguistic objects, in any structural description, are in fact very few with respect to the theoretical possibilities (Epstein et al. 1998); given a simplified structural description of a sentence like the one below, we can provide plenty of relations among elements that are completely irrelevant in terms of generalizations (e.g. John is two-nodes-far from Mary):

(47)

\[
\begin{align*}
& X \\
& \quad \downarrow \text{likes} \\
& \quad \downarrow \text{John} \\
& \quad \downarrow \text{Mary}
\end{align*}
\]

Among the significant ones, two are usually assumed to be necessary (and sufficient) in order to describe tree-structures: precedence and dominance

**Precedence** is a total (that is, defined on all the elements of a given set) strict (asymmetric, irreflexive and transitive) order defined on terminal nodes (leaves) of the tree: {“John” precedes “likes”, “likes” precedes “Mary”, “John” precedes “Mary”} or in a more compact way: <John, likes, Mary>.

---


38 But see Frank and Vijay-Shanker 2001 for different primitives.
*Dominance* is a partial order (even though “locally total”, Kayne 1994), asymmetric, transitive and reflexive (assuming that every node dominates itself) defined on the whole set of nodes of the tree: {“X” dominates “John”, “X” dominates “Y”, “X” dominates “likes”, “X” dominates “Mary”, “Y” dominates “likes”, “Y” dominates “Mary”, any-node dominates itself... } otherwise: {X < John, X < Y, Y < likes, Y < Mary}. Another important (derived, in the sense of Gorrell 1995) relation is \( C(\text{onstituent}) - \text{Command} \). This relation has been employed to define government and binding conditions, scope constraints (Cf. §2) and proposed in many flavors:

(48)

a. Reinhart 1979

\[ A \text{ C-commands } B \text{ iff:} \]

the first branching node dominating \( A \) dominates \( B \), \( A \) does not dominate \( B \), and \( A \neq B \).

b. Chomsky 1986b

\[ A \text{ C-commands } B \text{ iff } A \text{ does not dominate } B \text{ and every } C \text{ that dominates } A \text{ dominates } B. \]

c. Kayne 1994

\[ A \text{ C-commands } B \text{ iff } A \text{ and } B \text{ are categories and } A \text{ excludes } B \text{ and every category that dominates } A \text{ dominates } B. \]

d. Chomsky 2000

\[ A \text{ C-commands } B \text{ iff } B \text{ is contained in the sister of } A. \]

e. Epstein et al. 1998

\[ A \text{ derivationally C-commands } B \text{ when:} \]

\( A \) and \( B \) are directly paired/concatenated by merge or by move,

in this configuration even \( A \) and \( B \) \( \text{C-command} \) each other, and \( A \)

asymmetrically \( C \)-commands all and only the terms of the \( B \)

with which \( A \) was paired(concatenated by Merge or by Move in the course of the derivation.
The non-primitive nature of these relations seems quite evident: Reinhart uses the notion of first branching node and dominance, Chomsky refers either to dominance or to sisterhood, Kayne incorporates notions such as exclusion and dominance, while Epstein et al. employ the concepts of Merge, Move and the idea of derivation. Even though these definitions are not equivalent, they all capture the very same set of essential phenomena (binding, scope asymmetries, movement and control etc.). It seems important to me to draw a clear distinction between (48).a-d and (48).e at least, since the first four definitions present a view of the C-command relation that is representational, while the last one can be thought of as derivational. This opposition, sometimes considered simply ephemeral, has indeed crucial implications for a computational model (as it will be explained in §2.2). For the time being, we should note that the main distinction lies on the assumption that representational relations can be defined among any node of the tree, simply depending on the overall geometry of the structure, while derivational relations are tightly related to the time the linguistic objects enter the computation. From a purely theoretical point of view, the derivational option is (potentially) more restrictive even though the description of (derivational) C-command provided in (42.e) does not take advantage of this opportunity, since it can access any object introduced in the computation up to the moment this relation is established. This “transparency” of the whole structure makes quite useless the derivational machinery. I would suggest that this problem is partially related to the bottom-up perspective the derivation is assumed to follow, and partially to the absence of phases (Chomsky 1999 or barriers, Chomsky 1986b). Another important fact we should examine is that all the definitions of C-command associate the domain of the relation to the success of an immediate constituency formation: the first branching node is the immediate result of the concatenation operation, while sisterhood is the relation between constituents that are concatenated; finally, Merge/Move are concatenating operations. Going back to the
first relations described in this paragraph, notice that a narrower version of the
dominance relation (e.g. immediate dominance, that is, dominance without
transitivity) would exactly entails the relation among constituents and their mother.
This might be not a coincidence.

1.4 Complexity Theory
In the last paragraph (§1.3), I pointed out that having a transparent representation
of the phrase structure leads to a bigger number of theoretically possible relations
among elements; this number is limited in a derivational approach that postulates
“opaque” structural boundaries. This point can be explained with a simple example
comparing two structural representations of the same sentence <a, b, c, d, e>:

(49)  

\[
\begin{array}{c}
\text{(a) } a \quad b \\
\text{(b) } A \\
\text{A} \quad B \\
\text{A} \quad C \\
\text{a} \quad B \\
\text{b} \quad C \\
\text{c} \quad D \\
\text{d} \quad e \\
\end{array}
\]

The difference between these representations can be accounted for in terms of
number of C-command relations (assume, for instance, the definition given in
(48).a) we can define among elements: in (49).a we can define eight relations taking
a as argument (a c-commands B, B c-commands a, a c-commands b, a c-commands
C, a c-commands e, a c-commands D, a c-commands d, a c-commands e) while in
(49).b, only the first four relations (a c-commands B, B c-commands a, a c-
commands b, a c-commands C) are directly derivable. In which sense this is
empirically an interesting result will be evaluated in the next chapter (§2.2.2, §2.3.3
and §3.4), it is however important to notice that this “impenetrability condition” has
been obtained postulating tree-splitting; this operation directly follows from the hypothesis that once processed, part of a tree becomes inaccessible to further elaborations (then also unavailable for the definition of new relations).

Which tree must be processed before is, by now, irrelevant, the important thing is that having fewer relations to be evaluated reduces the problem complexity.

Even though this fact seems pretty intuitive, the definition of complexity of a problem is indeed a precise, formal notion that requires an accurate definition: following Papadimitriou (1994), the complexity of a problem can be defined as a function of the resources, essentially time and space, needed to solve the problem, where time represents the number of steps needed to reach the solution and space is the memory required to store/retrieve the information to be elaborated.

We can say that the complexity of a problem is proportional to the size of this problem, which is determined essentially by three factors:

i. the length of the input ($n$);

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

iii. the algorithm used to explore this space.

In order to predict the tractability of a problem (namely to guess whether or not this problem has a solution and how much it will take to discover it), we are interested in the growing rate of the complexity function, that is, roughly, how many more steps we shall make to find the solution, any time we add an extra item to the input. Without going into much details, a problem with a complexity linear function\(^{39}\) would be tractable since any extra item in input would require (at worst) a fixed

\[^{39}\text{e.g. } f = cn, \text{ where } c \text{ is a constant and } n \text{ the length of the input would have a linear order of complexity, namely } O(n).\]
number of extra steps to solve the problem; also polynomial functions are tractable\textsuperscript{40}. Problems arise when the order of the function is exponential or factorial (O(n\textsuperscript{n}), O(n\textsuperscript{n!})): the growing rate of these functions is so fast that there are no possibilities to assure a solution in a reasonable time\textsuperscript{41}. In short, generally, we should avoid exponential or factorial complexity functions since fingerprints of intractable problems. Within this context, it is however appropriate asking why we should calculate the complexity function in terms of input length when, in fact, a derivational perspective would allow us chunking this input in finite, manageable pieces. Actually, while for the moment there is little to say about the third point (the algorithm to be used will be presented in chapter 3), both the length of the input and the space of the problem are drastically reduced by the derivational trick presented in (49).b; this is essentially due to the combinatorial progression that regulates the growth of the number of relations definable on a given input set: the number of relations, such as c-command, has a growth that is polynomial with respect to the introduction of a new terminal item in a (binary branching) tree structure; given \( n \) terminals, in fact, the number of c-command relations is exactly \( n^2-n \) (e.g. with 3 terminals we can define 6 relations, with 4 terminals 12 relations, with 5 terminals 20 and so on). Then the tree-splitting proposed in (49).b does not simply have the effect of reducing the number of relation involving the terminal \( a \), but the overall number of relations from 20 (since we have 5 terminals) to 12 (two times 6 relations, that are the number of relations defined on a tree of 3 terminals). This is not an impressive result, since, as we mentioned before, problems with polynomial order of complexity would have been tractable, but it is however an interesting reduction.

\textsuperscript{40} e.g. \( f = cn^2 \), would have a complexity order = \( O(n^2) \); then, assume \( c=1 \), with 5 items in input we should make at worst 25 steps; with 6 items, 36 steps and so on.

\textsuperscript{41} See Barton and al. 1987 for more details.
As it will be clear later on, better results will be obtained formalizing the idea of cyclic movement and the notion of phase (Chomsky 1999-2000) with respect to specific problems (i.e. ambiguity and long distance dependencies, cf. §3.4).

As Chomsky points out (Chomsky 2000:111), the fact that this computational complexity calculus should matter for a cognitive system is essentially an empirical matter. To my opinion, this would imply having a formal complexity measure of the linguistic processing that we could compare with genuine human processing data.

Unfortunately, a complexity function expressed in terms of input length obfuscates an essential property of the linguistic processing system, namely its independence from the bare number of tokens to be processed, as expressed by the following contrast:

(50) a. "The reporter who the senator who John met attacked disliked the editor.
   b. The editor was disliked by the reporter who was attacked by the senator who John met.

(50).b is longer than (50).a but “easier” to parse. Intuitively, in order to be “empirically adequate”, a complexity theory should predict that what is “easier” for humans should be “easier” for the algorithm too, then the “simplicity” of a sentence should be productively expressible by an adequate complexity function. Gibson (1998), among others, suggests that a “cost function” in parsing should be sensitive to the input length in a relative way: namely new words should have a cost of structural integration based on the actual distance, calculated in terms of some relevant elements (new discourse referents, in Gibson’s theory), that intervene between the word and its integration point (namely where it is re-merged):
In (51).a both the “paths” $a$ and $b$ have a cost equals to 2 (because “John” and “met”, that, within Gibson’s theory, are new discourse referents, are crossed), while in (51).b only $a$ has a cost. This should be enough to understand what causes the contrast.

In addition, Gibson suggests that another part of the function cost is determined by what he dubbed the storage cost, that is, every syntactic head, predicted at some point as the minimum requirement to complete in a grammatical way the sentence, has a cost:

\begin{align*}
(52) \quad & \text{a. } \# \ldots \text{the senator who John met attacked } \\
& \quad \begin{array}{c}
2 \quad 1 \\
3 \quad 2 \quad 1 \quad 1
\end{array} \\
& \text{b. } \ldots \text{attacked by the senator who John met } \\
& \quad \begin{array}{c}
1 \quad 2 \quad 1 \quad 0 \\
2 \quad 1 \quad 0
\end{array}
\end{align*}

I think the way he decomposes complexity function in two relevant components (the structural integration cost and the storage cost) is fairly interesting, since it allows us to get rid of the notion of bare input length in a meaningful way. For instance, recalling the discussion in §1.2, these two components could represent the cost of keeping an active pattern (that is, an element to be re-merged should be stored in a memory buffer; cf. storage cost) plus the cost of introducing new (top-down) expectations once processed a lexical head with CSel requirements, or a functional element before its head (that is a way to explain the structural integration cost). Note that, theoretically, an interesting configuration could be obtained cyclically: namely, when all the expectations in the domain of a lexical head are satisfied; this
should be the case of the thematic role saturation of the verb (and noun). I would like to suggest that these special moments of equilibrium correspond to phase boundaries in Chomsky’s terms (Chomsky 1999, cf. §2.2.2, §3.4.3).
Formalizing a grammar forces us to specify anything we (minimally) need in order to describe crucial properties of language. This effort reveals an extremely fuzzy border between what has been usually considered competence and what has been dubbed performance even though the classical distinction (Chomsky 1965) between grammaticality and acceptability seems sound (§2.1); this is because, formally we could easily include in our competence a formalized constraint that another theory consider part of our performance (e.g. memory limits).

Bearing this in mind, here I want to discuss a specific formalization, explicitly considering to what extent it encodes linguistic intuitions and at what computational cost it does. In order to evaluate these two aspects, both representational (Extended Standard Theory, §2.2.1 and Cartographic Approach, §2.2.4) and derivational (Minimalist Program, §2.2.2 and Phillips’ 1996 model §2.2.3) linguistic theories will be reviewed (§2.2). Eventually, their relative computational coherence/efficiency will be evaluated (§2.3): first a principle-based approach (§2.3.1), then Stabler’s 1997 formalization of Chomsky 1995 Minimalist Grammar, finally an implementation of the derivational engine in terms of phases, probes and goals sketched in Chomsky 1999-2001, due to Fong 2004 (§2.3.3).

2.1 Between competence and performance: processing models

It is necessary for both generative and computational linguistics to define precisely the linguistic knowledge representation; trying to take into account those properties that make this representation cognitively plausible is a plus that would make our theory explanatory more adequate.

From a formal point of view, we think of a language as an infinite set of sentences, each of them associated to at least one Structural Description (SD) that makes
explicit some relevant relations among elements (such as linear order, relative scope, thematic/eventive structure etc.). From this perspective, the competence is the generative (that is formal) procedure that allows any native speaker to recognize and produce the whole infinite set of grammatical sentences that constitute his/her language.

The competence is an intensional procedure, rather than an extensional one: it would be impossible to store in our brain the infinite number of grammatical sentences (with their SDs), but, in fact, we definitely produce/understand completely original and grammatical linguistic expressions. In this sense, a generative grammar is a formal description of this intensional procedure (Chomsky 1986a).

The real use that any speaker/hearer makes of this knowledge to produce/understand sentences seems to be a different matter: memory limitation, restrictions in computational resources accessibility, rapid fading of signals are only some of what have been usually considered performance factors. These “extra-linguistic” factors somehow limit the use of our linguistic knowledge and are alleged to be responsible, for instance, for the low acceptability (that is different from ungrammaticality) of nested constructions (1).a or self-embedding (1).b, (Chomsky 1965:10):

1. a. I called the man who wrote the book that you told me about up
   (I called up the man who wrote the book that you told me about)
   
   b. The man who the boy who the students recognized pointed out is a friend of mine
   (The students recognized the boy who pointed out the man who is a friend of mine)

Therefore, acceptability is a matter of language use (namely performance) while grammaticality is well-formedness with respect to our grammatical knowledge (that is competence).
It seems clear, from this picture, that the competence is inscrutable if not through performance data.

An absolute distinction between competence and performance could be drawn only on the basis of a set of arbitrary stipulations concerning what performance factors are about:

2. Competence (C) + Performance (P) factors = Linguistic Behavior (LB)

In fact, all of the following possibilities are equally plausible and essentially equivalent from a purely formal (that is, generative) point of view (c₁ is a competence factor while pₙ a performance one):

3. a. C{c₁, c₂, ..., cₙ} + P{p₁, p₂, ..., pₙ} = LB
   b. C{c₂, ..., cₙ} + P{p₁, p₂, ..., pₙ, c₁} = LB
   c. C{c₁, c₂, ..., cₙ, p₁} + P{p₂, ..., pₙ} = LB

This is pretty straightforward in a modular (additive) system (cf. §2.2.1, §2.3.1): for instance, memory limitation could be either a performance factor or a competence filter that imposes extra constraints on the generative power of the grammar in a way that is essentially similar to other competence principles (such as conditions on pronominal binding). It has been usually assumed, at least before the beginning of the Minimalist inquiry (§2.2.1), that a model like the one presented below could be fairly representative of the competence/performance dualism:
Psycholinguistics is the research field that deals with performance factors, evaluating plausible models for the “grammatical basis of linguistic performance” (Berwick and Weinberg 1986): keeping into account the whole linguistic behavior in order either to confirm or to disconfirm generative theories about competence, besides native speakers’ grammaticality judgments, two major complementary sources of information to be taken into account seem to be sentence processing (real-time elaboration of structural descriptions during perception of visual or auditory linguistic inputs) and language production (the process involving a conversion from non-linguistic conceptual intentions to linguistic expressions).
Assuming that flexibility and realism\textsuperscript{42} are actual properties of our linguistic competence, these psycholinguistic investigations should converge toward the very same competence model, which should interface with at least two performance systems (Chomsky 1995): a conceptual-intentional one (which we know very little about) and a sensory-motor one.

In the early eighties, a similar problem was couched within the discussion about the relation between human grammar and parser: how could the parser use the grammar to perform human-like tasks in sentence processing? According to Berwick and Weinberg (1986) there are at least three relevant classes of relations between the parser and the grammar:

i. \textit{token transparency} (Miller and Chomsky 1963) – every rule/principle postulated by the grammar is mirrored by a parsing step (null hypothesis);

ii. \textit{type transparency} (Bresnan 1978) – the parser essentially mimics the grammatical rules/principles but in a way that only preserves the typological nature of principles/rules (namely the goals are the same even if the means to attain them may vary), implementing the processing functions in a way that is (more) psycholinguistically plausible;

iii. \textit{covering grammar} (Berwick and Weinberg 1986) – the relation between grammar and parser is only expressed in terms of an equivalence relation with respect to the set of linguistic phenomena captured; this can be realized by completely independent procedures/principles/rules from the two different perspectives. This seemed to be the best way to keep

---

\textsuperscript{42} In the sense explored in the introduction of this book (points I and IV): \textit{flexibility} implies that the same grammatical knowledge should be used both in \textit{parsing} and in \textit{generation}; \textit{realism} requires a faithful reproduction of productive phenomena involved in \textit{comprehension} and \textit{production} accounting for complexity issues.
separated descriptive/explanatory adequacy issues (domain of the grammar) from efficiency ones (domain of the parser implementation).

The third option clearly allows for more freedom and it has been probably the most successfully pursued at the time of Government and Binding (GB) approach (§2.2.1): computational implementations that used this grammatical framework (for instance Principle-Based Parsing, §2.3.1) had to solve many “efficiency” problems due to various underspecified aspects of the grammar, like ordering principles/rules so as to limit the number of ill-formed structures to be examined (Fong 1991) or (partial) pre-compilation of the rules to speed-up the parser (Merlo 1996).

What is common among principles formalization using first order logic, efficient LR tables and human use of the linguistic competence could hardly be something more than a covering relation.

This problem slowly faded out during the last decade, probably since the generative mainstream became more aware of some fundamental limitations of the GB framework, moving toward cognitively more plausible theories of competence: the Minimalist Framework, for instance, tried to consider some “performance issues” such as interface conditions, the notion of spell-out, the idea of cyclic derivation (by phases) during the phrase building operations (see §2.2.2). Some of these new devices try to account for problems which were classically considered “performance factors”, like the cost of accessing the lexicon (which led to the concept of Numeration), global/local economy conditions (any operation has a “cost” then, for instance, Merge preempts Move) or memory limitation (the elements are “active” in structure building only during a relevant period of time, the phase; after the phase is completed, its components are largely inaccessible to further operations).

These modifications make the original distinction between “declarative properties of the syntactic theory” and “procedural notions” (Crocker 1996) not easily deducible from the linguistic theory, and the classical grammar-parser opposition became less
straightforward. Generally, the performance side of the linguistic behavior became more penetrable to theoretical incursions (see the Twenty-Fifth Annual CUNY Conference on Human Sentence Processing proceedings, Bradley et al. 2012).

Thus, with relatively few departures from “orthodox minimalism” (§2.2), the token transparency idea reappears as a real option. One of the clearest pictures from this perspective is Phillips’ book (1996). Phillips presents many empirical arguments supporting the assumption that the distinction between grammar and parser is unnecessary, ending up with the following model (Parser Is Grammar, PIG):

(5) PIG model (Phillips 1996:255)

This clashes with the standard model assumed within more classic frameworks such as (4), where lexicon, grammar and parser have been considered “modules”, subject to independent principles/rules.

Although this approach is exactly an instantiation of the problem outlined in (3) (then it could easily turn out to be equivalent to a model that embeds in the same “grammatical box” resources such as working memory, past experience etc.43) what crucially represents a breakdown with the generative linguistic tradition is the

43 A cursory glance to this option could suggest that this unification has not been seriously attempted yet, just because the “modules” under the “resources” box are extremely difficult to formalize even if intuitively graspable.
assumption that classical performance phenomena (like ambiguous structures resolutions, §2.2.3) should be accounted for in terms of competence factors, namely they represent clues about how our linguistic knowledge is structured and not just how it is used.

We should keep this idea in mind when we will discuss the “virtual conceptual necessities” (Chomsky 2000:111, §2.2.2) that any linguistic theory should postulate in order to be flexible and realistic (in addition to explanatory and universal).

2.2 Derivations or representations? some linguistic models

From a psycholinguistic perspective, it seems fairly accepted that parsing and generating sentences are incremental processes. Since building phrase structures piecemeal requires a precise sequence of steps, the dynamic nature of these processes justifies the idea of derivation, that is, the output of the computation is reached only after a sequence of successive, concatenated operations.

Generative grammars both historically (Chomsky 1957) and more recently (Minimalist Program, Chomsky 1993-2001, §2.2.2) adopted the derivational perspective for structure building purposes.

Note, however, that this idea of derivation is often unlinked to the processing side (Phillips’ model, §2.2.3, is a notable exception): Chomsky’s Minimalist Program (Chomsky 1995:223,fn3), for instance, does not entail any real temporal sequence in the application of the operations: a derivation has to be intended as a sequence of purely formal, then abstract, successive transformations that operate on SDs. Following the considerations presented in the previous paragraphs, this is however an unwanted complication in computational/cognitive terms: processing inferences should be both reasonable and desirable

Economy considerations and ambiguities resolution strategies has been shown to lead to a cognitively plausible (quasi-)deterministic processing system such as the crash-proof syntax

---

44 Economy considerations and ambiguities resolution strategies has been shown to lead to a cognitively plausible (quasi-)deterministic processing system such as the crash-proof syntax
On the other hand, there are linguistic frameworks (Government and Binding Approach, Chomsky 1981-1986b, §2.2.1, Cartographic Approach, Cinque 1999, Rizzi 1997 §2.2.4) where the derivation of a SD is completely irrelevant: any order of application of principles/rules would bring to the very same geometrical configuration among elements.

Roughly speaking, while the first models (derivational) entail a sequence of discrete steps such that each step produces an output that is the input of the next step, the second ones (representational) only specify requisites of well-formedness, without any specification of what has to happen before and what after (Jackendoff 1997:12).

To better understand these different positions let us concentrate on a crucial point of discussion: the chain Vs. movement opposition (of interest for our purposes, since it represents the most classical case of Long Distance Dependency).

From a derivational perspective, movement really implies the displacement of an element from the base position (structurally the lowest position where the element occurs) higher up in the phrase structure⁴⁵, leaving behind a trace that is assumed to be a perfect copy, even though phonologically null, of the moved object (copy theory of movement, Chomsky 1995-2001). Following the minimalist trend (§2.2.2), any movement is triggered by feature requirements that have to be satisfied by successive cyclic operations targeting only the closest positions where these requirements can be satisfied (shortest move, Chomsky 1995).

model (Frampton and Gutmann 2002) where no attempt to build unused SDs is made; This is an interesting hypothesis to be evaluated in computational terms that would allow us to make an efficient use of computational resources (a similar attempt has been already explored in Marcus 1981, among others, but only from a parsing perspective).

⁴⁵ Any movement is assumed to be toward c-commanding positions, but see Richards’ (2004) analysis of wh-movement in Bulgarian or the classical analysis of Right Node Raising (Postal 1974).
Watching the very same phenomenon from a representational perspective, the moved element \( x \) and its traces \( t \) form a chain \( \langle x, t_{x,n}, t_{x,n-1}, \ldots, t_{x,0} \rangle \) for which specific conditions hold:

- \( x \) C-commands \( t \) in the chain and \( t_{x,j} \) C-commands \( t_{x,j} \) where \( j < i \)
- between \( x \) and \( t_{x,0} \) there are no intervening element \( y \) such that \( x \) and \( y \) belong to the same class.

These two positions have been considered as incompatible options\(^{46}\), illusory alternatives\(^{47}\), or equivalent devices\(^{48}\).

Instead of taking sides on this discussion, it is worth trying to go deeper in this distinction, defining some theoretical differences that might be useful not just to classify approaches, but to understand their essential properties and to evaluate seriously what can be represented and what should be derived.

(6) some differences between representational and derivational frameworks:

i. **completeness** of the Structural Descriptions (SDs)
   - representational approaches keep unified and static the SDs of the linguistic objects at any level of representation (if multiple levels are considered as in standard transformational grammars); the whole computational process results in a set of complete multiple SDs expressing the whole structure of the sentence from different perspectives;
   - derivational approaches transform SDs at every application of any principle/rule; the result of the computation is a single representation, if

---

\(^{46}\) Epstein and al. 1998.

\(^{47}\) Pure derivational systems cannot exist, they need representations to work, then they should be defined at best weakly derivational, Brody 2002.

\(^{48}\) For a formal discussion on the equivalence between derivations and representation, see Michaelis 1998.
any, that is the history of the whole derivation. Otherwise, during the computation any SD reflects only a partial Structural Description of the whole sentence.

ii. ordering principle/rules

- representational approaches make no explicit assumptions about the application of principles/rules. All combinations should result in the same SD.

- derivational approaches postulate an order (sometimes interpreted as the emergent property of a cost function, Collins 1997) in any operation that results in a grammatical (that is legal) transformation of the SD.

iii. nature of the relation among elements

- representational approaches define the relation among elements in a static way, using properties of complete SDs (e.g. C-command can be defined between any elements A and B present in the SDs; this relation will be always valid within the same SD, like the elements in a chain);

- derivational approaches determine the relation among elements in a dynamic way, crucially related to the time they enter the computation (e.g. A enters a relevant relation with B at the time τ); then the object (status) changes in the course of the derivation (e.g. the function f takes two object A and B and replaces them with C) so that the previous relation becomes inaccessible to the next steps.

iv. nature of the constraints

- representational approaches should postulate filters, namely procedures that select only grammatical SDs, once a set of correct SDs is produced by the indiscriminate application of principles/rules;
- derivational approaches can constrain the derivation, preventing it from producing ungrammatical outputs at any application of any principle/rule (e.g. by using economy conditions).

v. processing implications

- representational approaches makes no explicit assumption on how linguistic competence is put to use (e.g. they stay completely agnostic with respect to flexibility and to realism issues presented in the introduction);
- derivational approaches entail processing expectations (either purely formal, Chomsky 1995, or psychologically plausible, Phillips 1996), given that they make crucial assumptions about the time the elements enter the computation (so flexibility and realism could be at issue).

This distinction is probably too strong to be used for categorizing most of the main generative frameworks, but it has the advantage of making clear some crucial differences between the two hypothetical theories of movement as summarized in the table below:
<table>
<thead>
<tr>
<th>i. completeness of SDs</th>
<th>representational</th>
<th>derivational</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>unique and complete:</strong> all instances present in the sentence appear in the chain:</td>
<td>&lt;x, tx₀, tx₁, ... txₙ₋₁, txₙ&gt;</td>
<td><strong>partial:</strong> only the relevant element (a segment, at best) of the “chain” is accessed at any step:</td>
</tr>
<tr>
<td>step 1: x;</td>
<td>step 2: tx₀ (x = tx₀);</td>
<td></td>
</tr>
</tbody>
</table>
| ... | ...
| step n: txₙ (txₙ₋₁ = txₙ); |

<table>
<thead>
<tr>
<th>ii. principle/rules ordering</th>
<th>representational</th>
<th>derivational</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>irrelevant:</strong> any order would postulate the same traces and discard ungrammatical options</td>
<td></td>
<td><strong>strictly defined:</strong> unless we define extra backtracking options, postulating a wrong movement would prevent the derivation from retrieving correct SDs</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>iii. relation among elements</th>
<th>representational</th>
<th>derivational</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>absolute:</strong> any relational property among elements in the chain is valid within a single SD</td>
<td></td>
<td><strong>relative:</strong> any relational property is valid only within a relevant lapse of time τₙ (at τₙ: &lt;txₙ₀, txₙ₋₁ₙ&gt;), then further operation (valid at τₙ₋₁) would not have access anymore to the single constituents that established this relation.</td>
</tr>
<tr>
<td>iv. nature of the constraints</td>
<td><em>filters</em> on the unique resulting representation (e.g. <em>case filter</em>)</td>
<td><em>constraints</em> on operation application (such as <em>shortest move</em>)</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>v. processing implications</td>
<td><em>none</em></td>
<td><em>rigid order predicted</em> (potentially, this could have direct implications for processing)</td>
</tr>
</tbody>
</table>

I will be back on this distinction in §3, showing how these properties make a difference in terms of computational complexity eventually arguing that the derivational perspective will be more economical as grammatical model.

Brody (2002), for example, points out that any derivational theory is at best weakly representational: he observes that in order to move an object from a constituent that contains it, this constituent has to be transparent for later operations (namely the representation uses some sort of SD as is was supposed to be the case only in representational approaches, cf. (6).iii).

This argument, clearly pertinent from a general point of view, is however irrelevant under the derivational perspective discussed in (6).iii, since constituents are not accessible anymore after they successfully built a certain phrase\(^{49}\).

---

\(^{49}\) The solution proposed to account for *movement* without any necessity of inspecting constituents in a previously built phrase will be given in §3 and it resides on the directionality of the operation (*top-down, from left to right*, as I will assume in the next chapter, *Vs. bottom-to-top* as Chomsky and Brody do).
In the next paragraphs I will explore two paradigmatic examples of what we usually consider representational and derivational frameworks: the Extended Standard Theory (EST model, Chomsky 1973-1986a) and the Minimalist Framework (Chomsky 1993-2001). Then an even more radically derivational version of the minimalist model will be presented following Phillips’ idea (§2.2.3). Finally, in §2.2.4 I will present a second (mainly representational) framework, the Cartographic Approach, useful to highlight an important complementary problem for any theory of movement: the definition of locality constraints.

2.2.1 Extended Standard Theory (EST) and Government and Binding (GB)

The Extended Standard Theory (EST, Chomsky 1973, 1981-1986a) is a theory of grammar well representative of the representational class; it postulates five levels of representation: a Lexicon, a Deep-Structure (DS), a Surface-Structure (SS), a Phonetic Form (PF) and a Logical Form (LF). These are levels of representation in the following sense:

- they are defined as sets of symbols;
- they describe their objects in terms of static Structural Descriptions (SDs);
- they require mapping algorithms to translate one SD into another.

The relation among these levels can be described as follow:
(7) Level structure in Extended Standard Theory (EST, Chomsky 1973-86)

\[ DS \quad \longrightarrow \quad Lexicon \]

\[ DS \quad \downarrow \quad SS \]

\[ SS \quad \rightarrow \quad LF \quad \quad PF \]

Logical Form    Phonetic Form

The rationale behind such a multi-level representation was mainly empirical:

\( DS \) was intended to isolate the domain where thematic relations took place and where lexical items were inserted in the process:

(8) \( DS \)

\[ \text{John} \quad \text{read} \quad \text{a book} \]

\( SS \) was the locus of most of the movement operations:

(9) \( DS \): did John read what

\( SS \): What, did John read it?

\( LF \) was where covert movements, relevant for interpretational purpose, took place:

(10) \( SS \): Every boy likes a book

\( DS1 \) \( (surface \ scope, \ \text{“there is a different book for every boy”}) \):

\[ \forall \text{boy}, \exists \text{book} : \text{likes(book, boy)} \]

\( DS2 \) \( (inverse \ scope, \ existential \ quantifier \ raised, \ \text{“there is a single book for every boy”}) \):

\[ \exists \text{book}, \forall \text{boy} : \text{likes(book, boy)} \]
PF accounted for phonological phenomena like head incorporation or cliticization. The EST model has been historically tied to the Government and Binding (GB) approach (Chomsky 1981-86). This framework tried to get rid of the numerous and complex rewriting rules that pervaded transformational grammars at that time, replacing them by a small set of universal principles. An adequate interaction among these principles plus a bunch of parameterized options should have been sufficient to account for many complex phenomena in a cross-linguistic perspective (Principle and Parameters, Chomsky 1981) that would have required hundreds of complex rules to be captured. For instance, before GB, passivization (11).b, focalization (11).c and binding (11).d had to be captured each by a specific rule that could interact in a hardly predictable way with other rules (11).e nonetheless leaving many data unexplained (11).f. This quickly leaded to an extremely complex and language-specific grammar design. Then the idea was to postulate a compact set of general principles like $\theta$-theory, Move $\alpha$, Case Filter and Binding Theory, reported in (12), which capture the very same set of phenomena without any reference to special rules or to language specific properties other than a lexicon and a bunch of parameters.

(11)a. John called Mary

\begin{align*}
\text{b. Mary was called by John} & \quad \text{(passivization)} \\
\text{NP}^1 V^{+\text{tense}} \text{NP}^2 & \rightarrow \text{NP}^2 \text{be}^{+\text{tense}} \\
\text{} & \quad \text{V+past_participle (by NP}^1) \\
\text{c. MARY John called} & \quad \text{(focalization)} \\
\text{NP}^1 V \text{NP}^2 & \rightarrow \text{NP}^2 \text{NP}^1 V \\
\text{d. John, called him_{ij}} & \quad \text{(binding)} \\
\text{NP V pro} & \rightarrow \text{NP}_1 V \text{pro}_2 \\
\text{e. *JOHN Mary was called by} & \quad \text{(applying passivization +} \\
\text{focalization to b.)} \\
\text{f. He$_2$ was called by John$_1$} & \quad \text{(unpredictable by this rules set)}
\end{align*}
(12)  \( \theta \)-theory

i. every argument receives one and only one thematic role

ii. every thematic role is assigned to one and only one argument

Move \( \alpha \)

a category \( \alpha \) can be moved anytime anywhere

Free indexation

indices are freely assigned to categories in A(rgumental) position

Binding theory

condition A - An anaphor (e.g. himself) is bound in its binding domain\(^50\)

condition B - A pronominal (e.g him) is free in its binding domain

condition C - A referential expression (e.g John) is free

Case filter

any overt NP argument has to be case marked or associated with a case position

An interesting property we should notice is the highly modular design of this framework: every principle is, theoretically, independent from any other; their interaction is an emergent property.

Taking a closer look at the principles structure, we should observe that while some of them generate more structures than what they get as input (for instance Move \( \alpha \) and Free indexation), some other principles constrains this behavior by filtering out

\(^50\) The binding domain would be defined as the Minimal Governing Category (Chomsky 1981), namely the minimal phrasal projection containing the element, its governor and a subject accessible to this element. In order to approximate this complex definition, we could paraphrase “binding domain” as the “portion of the tree C-Commanded by a potential binder”.

88
unwanted solutions (the *case filter*, for instance, rules out any structure where even a single NP is not marked for case).

An important filter of this kind, introduced in §1.1.5 and widely used within the GB framework, is the *X-bar theory* (Chomsky 1970), which captures a productive generalization about the internal structure of any syntactic constituent, namely that all lexical categories seem to project the very same structural skeleton once introduced in the phrase structure, as depicted in (13.a):

(13)  

(a)  

```
XP
   
YP  X'

specifier

X°  ZP
```

(b)  

```
head  complement

NP  VP  AP

spec  N'  spec  V'  spec  A'  ...

N°  comp  V°  comp  A°  comp
```

c.

```
DP  TP  CP

spec  D'  spec  T'  spec  C'  ...

D°  comp  T°  comp  C°  comp
```
This seemed to be a tenable hypothesis not only for lexical elements (Nouns, Verbs, Adjectives (13).b), but also for functional ones (Determiners, Tenses, Complementizers (13).c, cf. Chomsky 1986a).

X-bar structures definitely played a crucial role in identifying empirically productive relations expressing cross-categorial similarities (as we mentioned in §1.1.5): the head-complement relation was the locus of the phrase selection, while spec-head relation is the structural environment for agreement; moreover the head-complement order could be parameterized, accounting for cross-linguistic variation (head-final languages, like Japanese Vs. head-initial ones, like English).

From a computational point of view, the X-bar schema also limited the generative power of the grammar, filtering out incompatible structures.

Summarizing, the following properties of the GB framework are then directly relevant for the present discussion:

i. the grammatical competence is described by a small number of universal principles plus some parameterized options that map lexical elements through SDs at distinct levels of representation;
ii. principles both generate and filter SDs;
iii. the order of application of generators and filters is irrelevant for the final SD.

Despite its power in terms of descriptive adequacy, this framework revealed critical flaws during the last two decades. For instance, the theoretical desideratum of identifying at each level of representation characteristic operations not allowed at other levels quickly turned out to be unrealistic: lexical insertion could happen both at DS and at SS (as tough-movement shows (14), Chomsky 1981, 1993); reconstruction possibilities showed that the domain of binding cannot be limited only to SS: since in (15).a the anaphor himself should be C-commanded by Bill to be coindexed with it, a possible solution, given the SS in (15).a, could be that binding
takes place at DS, before the wh- phrase is moved, but this would leave unaccounted for the Principle B violation in (15).b.

(14)  *John, is easy [CP PRO to please e]

(15)  a. [which picture of himself; ] did Bill; see t?
     b. [which picture of him<∅>]; did Bill; see t?

Moreover, two other critical problems are related to the free application of the principles: first of all, many emergent properties of this wild interaction turned out to be unpredictable and sometimes unwanted (cf. §2.3.1); second, generators like Move a or Free indexation appear to be computationally very expensive: if filters do not constrain as soon as possible generators output, SDs grow in number following easily a factorial progression, which is extremely hard to be computed real-time with finite resources as we have seen in §1.4.

2.2.2 The Minimalist Program

The Minimalist Program (Chomsky 1993-2001) seriously tackles some of these problems, preserving the Principles and Parameters setup. I wish to emphasize three aspects of this approach that are relevant for a computational model of the competence:

- the notion of perfection;
- the simplification of the core computational system;
- the derivational approach to the Structural Descriptions and the use of economy heuristics.

As explained in the previous paragraph, the proliferation of highly technical (often ad-hoc) solutions to limit the unwanted generalizations due to unconstrained principles application created an unmanageable tension between descriptive and explanatory adequacy. The solution proposed by Chomsky was then to remove unnecessary machinery, limiting the theoretical assumptions to the bare essential necessities. Following this logic, the null hypothesis was to consider the Faculty of
Language (FL) as a perfect organ, that is, the best solution for bridging sounds and meanings (the two performance systems that FL necessarily interfaces with). Perfection, in this sense, essentially means avoiding any extra symbol in representation and extra step in derivation, beyond the minimal conceptual necessities.

The strongest version of this idea takes the shape of the Strong Minimalist Thesis: it takes place at DS, before the wh- phrase is moved, but this would leave unaccounted for the Principle B violation in (15).b.

(16) **Strong Minimalist Thesis** (Chomsky 2001:3, SMT)

FL should be defined only in terms of Interface Conditions and general conditions of computational efficiency.

For instance, the distinction between **Surface** and **Deep Structure** must be discarded (because of its empirical inadequacy showed, for instance, by (14) and (15)) whereas the role of LF and PF must be “reduced” from levels of representation to interface conditions (or legibility conditions, Chomsky 2000:94), namely precise requirements the linguistic representations have to conform to before being shipped out to the performance systems. Minimally, these performance systems are two: the conceptual-intentional system (reminiscent of the LF level) and a sensorimotor system (roughly speaking, the old PF). They are assumed to be external to FL (even if they actually impose constraints to FL objects) in a very important way, namely they could respect different principles/organization (Chomsky, for instance, assumes the binding theory presented in (12), namely the “module” of the grammar

---

51 “FL can be regarded as a language organ, in the informal sense in which the visual system, the immune system, and the circulatory system are commonly described as organs of the body: not objects that can be removed leaving the rest intact, but subsystems of a more complex structure that we hope to understand by investigating parts that have distinctive characteristics, and their interaction”. Chomsky 2000:90.
responsible for the interpretation of pronouns, anaphoric and referential expression, to be part of the conceptual-intentional system rather than of Narrow Syntax). Thus FL would produce expression of the form:

\[(61) \quad \text{Exp} = \{\pi, \sigma\}\]

where \(\pi\) is a finite set of phonological features\(^{52}\) arranged (within a SD) in a way that is interpretable for PF, and \(\sigma\) is a finite set of semantic features arranged so as to be interpretable for LF.

Even X-bar structure, as presented in (13), has been revisited under the light of some disturbing phenomena, for example clitic movement: in their thematic position they behave as full XPs, but since they incorporate to verbal heads, they should be heads as well (Chomsky 1995:249). Another problem was related to asymmetric feature checking: Case seems to be assigned to the subject by the verb (or a verbal related functional projection), but it is the subject that determines agreement on the verb, not vice-versa; this asymmetric relation cannot be predicted simply in terms of spec-head relations (Chomsky 1995:258). Then categories within the Minimalist Program, are considered elementary constructions (bare phrase structures, Chomsky 2000:249), namely direct projections of the properties (features) of the lexical items, without any bar-level or lexical item/X°/X'/XP distinction. This would make it possible to avoid the introduction of extra symbols in the computation, meeting what has been called the Inclusiveness Condition:

\[(17) \text{Inclusiveness condition (Chomsky 1995:228)}\]

\[\text{any structure formed by the computation is constituted of elements already present in the lexical items selected for N[umeration]}\]

As we discussed in §1.1.2, Numeration refers to the operation of one-time selection of items from the lexicon that will then be available for the computation. Even

\(^{52}\) In the sense proposed in Chomsky 2001:5 fn.14.
though slightly revisited in the last decade, the main idea is still the same: accessing the lexicon has a computational cost; this cost would be too high if every time a new item should be picked up during the process; because of that, a fair assumption is to limit the access to the lexicon to only once, namely at the beginning of every relevant phase of processing\textsuperscript{53}. Accepting (62), nothing but features from the lexicon will become part of the SD at the interface levels. Essentially these features should be either semantic or phonological; there is evidence, however, that there are more abstract features, which are interpretable neither at PF nor at LF (such as case on DPs, agreement $\phi$-features\textsuperscript{54} on verbs etc.). These features are dubbed formal (in some version of the Minimalist Program even uninterpretable) and they must be removed by the computation before reaching the interfaces. This distinction is useful to understand the new conception of movement, since, following (17), GB traces have no room within this framework (since they are not present in the lexicon). Then a trace has to be redefined as a (perfect) copy of the moved element without its phonological features (Chomsky 2001:9):

(18)\textit{Copy theory of movement}

traces are perfect copies of the moved element, with no phonological features.

Moving toward the analysis of the (simplified) computational machinery, we should observe that the Minimalist program tends to reduce the whole transformational apparatus to a single, simple and universal operation that builds structures recursively. This is Merge (cf. §1.2.1):

\textsuperscript{53} The term \textit{phase} in this context is used in an informal way. See (23) for details on a more formal notion of \textit{phase}.

\textsuperscript{54} Person and number.
(19) *Merge* (adapted from Chomsky 2001:6)

is a no cost operation which takes two elements $A$ and $B$ (already
constructed) and creates a new one consisting of the two: $\{A, B\}$

As we mentioned in §1.2.1, early minimalist versions suggested that the form of the
new object built by Merge should have been something like $\gamma \{A, B\}$, where $\gamma$ is the
label of the newly formed element. Given that the label should be easily predictable
from the constituents that merge, either it is the product of a general rule (e.g. the
head of the constituent is the label) or it is superfluous (Collins 2001); having
something else would result in a violation of the Inclusiveness Condition.

This definition of merge is general enough to capture both simple combinations of
elements that appear to be adjacent (64') and even displacement cases (64''), in fact,
many scholars pointed out that the notion of move presents many similarities with
that of merge (see Kitahara 1994 and Starke 2001). Chomsky’s recent papers
incorporate this insight, keeping, however, a distinction that minimally discerns
these two relations (Chomsky 2001:8):

(19)' *External Merge*

\[ A \quad B \]

(19)'' *Internal Merge (old Move)*

\[ B \quad \quad \quad \quad A \]

...
Chomsky (2001:9) notices that argument insertion (base position) is only fed by External Merge, while “all the rest” (derived positions) can be satisfied by means of internal merge (e.g. EPP/OCC features\(^{55}\), left peripheral criteria, see §2.2.4).

Before Chomsky 2000, it was usually assumed that movement is a “last resort” to eliminate uninterpretable features and thus to meet interface conditions: an element \(A\) (for instance a subject) had to rise to merge with \(B\) (an inflected Tense head) to check \(B\) uninterpretable features (uninterpretable \(\phi\)-features on the verb). Later work inverts this perspective, looking at displacement as directly triggered by the element that bears interpretable features, the \textit{probe}, as soon as it enters the computation: the dynamics of displacement is triggered by this probe which, because of ability of verifying uninterpretable feature, it searches in its domain for a goal, namely an element with an uninterpretable feature that can be checked against the interpretable one. The relation that holds between the probe and the goal is an agree relation: these elements should match in features “under non-distinctness”, that is, either the features of \(A\) and \(B\) have the same values or any unvalued feature \(F\) of \(A\) is interpreted as having the value of the same feature \(F\) of \(B\). After a full checking operation, the moved element seems to be blocked in its landing site. To capture the freezing effect, Chomsky assumes that both the probe and the goal are \textit{active} as far as they do not check their features against each other. Then, a full check of all the uninterpretable features \textit{inactivates} both the probe and the goal, as shown by the impossibility of checking multiple case features as in (20).

(20) a. \textit{John said} \([TP he, [VP ti went to bed very early ]}\]
    b. *\textit{John said} \([TP ti, [VP ti went to bed very early ]}\]

\(^{55}\) EPP stands for \textit{Extended Projection Principle}, the requirement expressed by any clause of having a subject. OCC(urrence) simply marks the requirement of a head to enter in a \textit{Merge} relation with some OCC of \(a\), as in (19)*, by \textit{Internal Merge}, or by \textit{Expletive Insertion}.
Apparently, *Agree* can be a long distance relation (Chomsky 1999:12), since even a distant goal (*people/a man*) can satisfy a probe (an inflected Tense, *are/is*) merged to satisfy its OCC feature with a φ-incomplete expletive (there):

(21)

- a. People, are likely to be in the room
- b. There are likely to be people in the room
- c. There is likely to be a man in the room
- d. *There is likely to be people in the room

The relation that holds between the probe and the goal is however local in the sense that no potential goal can intervene without checking the features of the probe as shown below:

(22)

a. *There are likely to be a man in the room with the guests
b'. There is likely to be a man in the room with the guests

The third (and last) minimalist assumption that I wish to discuss is the nature of the derivation. This point is relevant to the discussion begun in §2.2, because:

- the resulting SD that has to meet the *Interface Conditions* is “complete” (in the sense of (6).i) only as history of the whole derivation, otherwise it is built piecemeal from bottom (innermost VP shell) to top (most peripheral CP);
- the operations apply cyclically and are ordered on the basis of the bottom-to-top building process: in order to build phrases we start from the heads, first merging their complements, then their specifier(s) and so on; any merge takes place at specific time, so that it makes sense to think in terms
of sequence of operations (e.g. thematic requirements are satisfied before OCC needs);
- any relation among elements is ephemeral, in the sense that it is present only during a relevant “phase” (a technical notion of phase will be formalized below, in (23)), then later stages of structure building can not have access to the elements within this “phase”;
- the whole computation, driven by feature checking, is potentially deterministic, since it is driven by economy considerations (i.e. the actual “cost” of each operation can be evaluated): this is an important difference with respect to earlier versions of the Minimalist Program; there should be no parallel exploration of multiple SDs, then comparing them on the basis of global economy considerations (total length of the derivation, total computational resources used); recent minimalist trends is to consider the derivations only driven by local economy consideration (Collins 1997, Chomsky 2000:99)

In this sense the Minimalist framework is mainly derivational, in the sense proposed in §2.2.

Summarizing these points, the derivational nature of this framework is justified by complexity considerations that deeply affect this approach (Chomsky 2000:104):
  i. Simple operations preempts more complex ones\(^{56}\);
  ii. the search space is limited (locality conditions, as in (22));
  iii. access to the features set F is restricted by Numeration;
  iv. the computation is locally determined (no look-ahead).

---

\(^{56}\) Merging \textit{there} is “less expensive” than moving \textit{a proof}:

i. \textit{there} is likely \([T_{def} t_i] \text{ to be a proof discovered}\]

ii. \textit{there} is likely \([T_{def} a \text{ proof to be discovered}]\]
Moreover, considerations on abstract “memory limitations” justify the assumption that as soon as an element is spelled out, there is no need to keep it in the working memory anymore. This is one of the leading reasons supporting the notion of phase:

(23) Phase

relevant unit during which the computation maps a subset LA' of elements selected in the Lexical Array (LA), to $< \pi, \sigma>$;

A phase PH has the form $[\alpha \{H \beta\}]$, where H is the head and $\alpha$ - H the edge of PH.

In this sense, a Phase identifies an object “relatively independent in terms of interface properties” (Chomsky 2000:106); so Phases are assumed to be CP (a complete discourse-related entity), vP (a complete predicative structure) and maybe also DP (a complete argument).

Rather than spelling out the whole phase, Chomsky (2001:5) proposes to spell out only $\beta$ while keeping the edge available for successive cyclic movement. The memory limitations I pointed out above are captured by the Phase Impenetrability Condition:

(24) Phase Impenetrability Condition (Chomsky 2001:5 PIC)

The domain of H is not accessible to operations, but only the edge of PH.

Quasi-algorithmically speaking (indeed many points remains crucially underspecified yet), we could sum up the whole computational procedure that FL should perform as follows:

(25) Minimalist computation

1. Select LA from the lexicon
2. Recursively maps LA to Exp by phases following this subroutine:
   i. select LA', such that LA' is a proper subset of LA, and kept it active in memory;
   ii. take the most embedded verbal head and merge it with the adequate complements in LA';
iii. look for OCC features on the head, then check them by internal merge if no lexical items in LA\(^i\) can satisfy them (following the economy preferences)

iv. proceed until LA\(_i\) is exhausted, then restart from i. until LA is exhausted too

This pseudo-algorithm can be illustrated by the following schema:


These assumptions largely unload on the lexicon the theoretical burden of accounting for cross-linguistic variation and apparent idiosyncrasies in phrase structures; because of that, Minimalism has been often considered a lexicalist hypothesis. A first fault of this approach is however clear: the lexicon is highly underspecified within Chomsky’s work. Another point that has been criticized is the assumption that a perfect design would imply no redundancy at all: Jackendoff (1997) points out that redundancy could be useful for learning purposes, then highly welcome if present. Moreover, interface requirements (especially at LF) are quite
underspecified, this only allows for highly arbitrary guesses concerning the status of bare output conditions.

2.2.3 Left to Right Incremental Processing

Phillips (1996), in a pre-phase era, makes an important step forward as introduced in §2.1: he convincingly shows that, because of constituency contrasts and ambiguities resolution preferences, the distinction between the parser and the grammar might be rather artificial; this suggests that the PIG model presented in (5) could be fairly adequate. Consider the following data:

(27) a. John [[[gives candy] to children] in their library]  
   a'. John intended to give candy to children in their library and [gives candy to children], he did it in their library  
   a".*John intended to give candy to children in their library and [to children in their library], he did give candy it  
   b. John said Bill [left yesterday]  
   b'. John [said [Bill left] yesterday]

(27).a-a" show that a normal right-branching structure (28).b would not predict the contrast in VP-fronting between (27).a' and (27).a"; this rather supports a left-branching structure as the one proposed in (28).a. On the contrary, the binding option (following (12)) would require a right-branching structure of the sort exemplified in (28).b where “children” C-commands “their”.

(28) a. give candy to children in their library  
   b. give candy to children in their library
Pesetsky (1995) postulates both structures (the layered structure (28).a as the locus where phrase extractions are computed, while the cascade structure (28).b as the locus of binding, negative polarity items interpretation and so on). Phillips proposes that a slight modification of (28).b would be sufficient to account for any apparent contrast that seems to require the left-branching structure proposed in (28).a. His solution is based on a left-to-right structure building derivation as illustrated below:

(29) a.  

\[ V \rightarrow N \rightarrow V \rightarrow V \rightarrow V \]  

b.  

\[ V \rightarrow V \rightarrow N \rightarrow V \rightarrow V \]  

c.  

\[ V \rightarrow V \rightarrow N \rightarrow V \rightarrow P \]  

d.  

\[ V \rightarrow N \rightarrow V \rightarrow V \rightarrow P \]  

\[ \rightarrow V \rightarrow P \rightarrow N \rightarrow V \rightarrow V \rightarrow P \rightarrow N \rightarrow V \rightarrow V \]  

\[ \rightarrow V \rightarrow P \rightarrow N \rightarrow \text{in} \rightarrow \text{their library} \]
The derivation in (29) (reminiscent of Larson’s (1988) VP shells) yields a solution to the constituency problem: in fact, both (29).b and (29).c represent temporary constituents (later destroyed by further merge operations) that are available for extraction at some specific point of the derivation. This is the cornerstone of the left-to-right structure building procedure Phillips proposes: temporary constituency is a property created during the derivation we could use to account for apparent paradoxes. Crucially, a left-to-right processing procedure creates temporary constituents that are different from the standard bottom-to-top\footnote{That is somehow different from bottom-up, that simply means starting from input data rather than from structure projections. On the other hand, bottom-to-top means starting from the inner verbal shell, then adding higher and higher layers piecemeal.} structure building proposed by Chomsky (see §2.2.2):

(30) The boy kissed Mary
   a. Phillips proposal:

   \[
   \begin{array}{c}
   \text{merge} \\
   \text{the} \quad \text{boy} \\
   \end{array} \quad \rightarrow \quad
   \begin{array}{c}
   \text{merge} \\
   \text{the} \quad \text{kissed} \quad \text{boy} \\
   \end{array} \quad \rightarrow \quad
   \begin{array}{c}
   \text{merge} \\
   \text{the} \quad \text{boy} \quad \text{kissed} \quad \text{Mary} \\
   \end{array}
   \]

   b. Standard minimalist proposal:

   \[
   \begin{array}{c}
   \text{merge} \\
   \text{kissed} \quad \text{Mary} \\
   \end{array} \quad \rightarrow \quad
   \begin{array}{c}
   \text{merge} \\
   \text{the} \quad \text{boy} \\
   \end{array} \quad \rightarrow \quad
   \begin{array}{c}
   \text{merge} \\
   \text{the} \quad \text{boy} \quad \text{kissed} \quad \text{Mary} \\
   \end{array}
   \]

For instance, only (30).a creates the temporary constituent [the boy kissed].
Since these kinds of constituents are relevant for phenomena such as those described in (27).a-a”, Phillips takes this to be a proof of the fact that the grammar incorporates a simple structure-building procedure, merge right:

(31) Merge right (Phillips 1996:18)

new items must be introduced at the right edge of the structure.

Even from a processing perspective, there seems to be a strong tendency to produce right-branching structures, privileging, moreover, the most local attachment, cf. (27).b-b”; this preference has been captured by the following principle:

(32) Branch right (Phillips 1996:19)

Metric: select the attachment that uses the shortest path(s) from the last item in the input to the current input item;

Reference set: all attachments of a new item that are compatible with a given interpretation

In fact, what Phillips’ model assumes is the incremental nature of the process without commitment to any performance modality. From a processing perspective, this seems to be fairly in line with psycholinguistic results: garden path sentences are abused but neat examples on this point, showing that words in input are integrated in the structure as soon as they are heard and that most of the time only one structural solution is pursued. This leads to “reanalysis” when an unexpected word is found as the next token of the input:

(33)a. The horse raced past the barn fell   (Bever 1970)

b. The horse (that was) raced past the barn fell (down)

The correct structural analysis requires the “reconstruction” of the missing words in (33).b, but, at the first reading, all native speakers encounter a parsing breakdown as soon as they reach the word fell; this is unexpected on the basis of the structural hypothesis adopted up to that point: “raced” is interpreted as the past tense of the verb “race” and “past” as the past tense of “pass”, missing the reduced relative and
the adverbial nature of “past” that, in fact, represents the only correct structural solution to integrate “fell” at the end of the sentence.

Garden path effects show that we do not pursue any possible structural hypothesis; rather, we make choices as soon as possible pursuing only one solution. The other options are retrieved only after a parsing breakdown, under the control of precise top-down expectations. This seems to be true not only of English (and in general of head-initial languages) but even of head-final languages like Japanese58.

2.2.4 The Cartographic approach and Locality Constraints on Movement


As I pointed out in §1.1.3, one of the clearest examples to grasp this idea is provided by Cinque’s analysis of adverbial positions (Cinque 1999): his work suggests that the IP shell is more structured that usually thought and, crucially, this seems to be a universal generalization supported by robust empirical evidence. In fact, even if many languages do not have specific morphemes that realize Cinque’s postulated functional heads, the existence of the latter is supported by the distribution of adverbial elements (allegedly occupying the specifier of these functional projections):

(34) a. John probably eats a lot
   a'. *John a lot eats probably

58 See Schneider 1999 for a review of the topic.
b. Necessarily John often sees Mary

b'. *Often John necessarily sees Mary

Probably and necessarily, in English as in many other languages, have to take scope respectively over a lot and often. Without trying to investigate the rationale behind this necessity, the best way to account for it is to assume, as we mentioned in §1.1.3, the existence of a hierarchy that we roughly expressed by these basic classes:

(35) modals > temporal > aspectual

This gross classification is refined in Cinque (1999) on the basis of distributional asymmetries such as the ones presented in (34). Being careful to discard apparent ordering counterexamples (adverbials that modify each other, focalization phenomena etc.) the resulting picture is quite articulated:

(36) Mood speech act Mood evaluative Mood evidential Mood epistemic T past T future

Mood 
Mood necessity Mood possibility Mood volitional Mood obligation

Mood 
Mood permission Asp habitual Asp repetitive I Asp frequentative I Asp celerative I

Asp perfecto Asp retrospective I Asp proximatives I Asp durative I Asp generic/progressive

Asp prospective I Asp sg completive I Asp pl completive I Voice Asp celerative II Asp repetitive

II Asp completive II T anterior Asp terminative Asp continuatives Asp frequentative II

Similar conclusions have been reached within other functional areas such as CP (Rizzi 1997, 2002-2004) and DP (Cinque 1994, Scott 1998). Subject (Cardinaletti 2002) and negation (Zanuttini 1997) positions have been investigated too, with similar fine grained results. The outcome of this research is then a detailed “map” of the functional phrase structure that would allow us to predict in a very precise way many universal distributional constraints.

This is an interesting result from the computational perspective adopted within this work, since it emphasizes the “sensitivity” of the movement options: since we understand the nature of the operations involving linguistic elements (and their features) even from their domain of applicability, it is worth considering locality.
conditions as the set of principled constraints used to narrow the range of application of these operations. Influential work on this issue is Rizzi’s *Relativized Minimality* (Rizzi 1990, 2002) and Starke’s feature hierarchy (Starke 2001) that I will quickly review in the rest of this paragraph.

As we noted before (§2.2.2) the application of the merge operation has to be local in a very narrow sense, namely two elements that enter a Merge relation have to be strictly adjacent. For movement the “local relation” has to be intended in a less restrictive way: the object and its deleted copy (or trace) could be relatively distant, unless some feature of the “same kind” as the one searched by the probe on a goal intervenes between them.

This is the insight sketched in (22) and repeated below:

(22) a. Probe Goal Goal

X

b. Probe Goal Goal

This simple picture, in fact, can predict in a very insightful way violations on extraction from weak islands (syntax), assimilations phenomena (phonology) and possibly even semantic and other perceptual (non-linguistic) facts (recall the discussion in §1.2).

Rizzi’s and Starke’s contribution to this idea has been to refine these locality conditions, relativizing them with respect to the classes of elements they apply to. We can draw some important generalizations from these works:

- locality is sensitive not to single features but to classes of features (Rizzi 1990-2004);
locality conditions are more predictive if they apply to a hierarchies of feature classes rather than to unorganized subsets (Starke 2001).

The first point is well explained by weak island violations: the movement of a wh-element is blocked not only when it crosses another wh-element as shown in (37),

(37) a. *how, do you wonder why I should cook this stuff t,?

but also when it crosses a negation (82.b), a focalized element (82.c), or a quantificational adverbial (82.d) (Starke 2001:5):

(38) b. *how, don’t you think that I should cook this stuff t,?

*how, do you think that I shouldn’t cook this stuff t,?

c. *how, do you think that, THIS STUFF, I should cook t, (not those eggplants over there)?

d. *how, should I often cook this stuff t,?

These facts circumscribe a natural class of sensitivity in terms of locality conditions:

(39) Q(uantificational) {wh, negation, focus, quantificational-adverbs}

All elements bearing one of these features are potentially deadly interveners for movement (or chain formation) of another element bearing a feature in the same class.

Argumental features represent another well known class of interveners (A-chains or A-movement):

(40) A rgumental {person, number, gender, case}

Starke suggests that these classes are not simply flat, but hierarchically organized: this is because hierarchical relations could be helpful to predict what has usually been caught as exception, but could, in fact, be perfectly integrated as an expected phenomenon given a richer feature structure. This fact can be represented by the example below:

(41) [which book], did who buy t,?
The chain *which book, ... who ... it* should represent a violation of the locality conditions, but, in fact, it is not: following standard interpretations, which book has “something more” with respect to other elements that usually cause a violation when extracted from a weak island. Even if the nature of the “extra property” of which book is not completely clear, the picture we can draw is pretty neat:

(42) a. *[α A] ... Bα ... [α A] *[wh, what] did [wh, who] buy <[wh, what]> ?

b. [αβ A] ... Bα ... [αβ A] [wh, d-linking which book], did [wh, who] buy <[wh, d-linking which book]> ?

c. *[α A] ... Bαβ ... [α A] *[wh, what] did [wh, d-linking which student] buy <[wh, what]> ?

d. [α A] ... Bγ ... [α A] [wh, what] did [animate John] buy <[wh, what]> ?

e. 

Given the hierarchy in (42).e the feature αβ and αγ are more specific than α. We could assume, following Starke, that locality conditions, in this hierarchy, apply horizontally (features in the same class potentially block each other, (42).a,d) and bottom-up (daughters are potential blockers for mothers, (42).c) but not top-down (daughters cross mothers without any locality violations, (42).b), i.e. a less specific feature cannot block a more specific one.
Then, following Rizzi (2004), the class of elements forming $A'$-chains can be refined using at least three subclasses as shown in (42). Otherwise, following Starke (2001), it can be structured as a hierarchy as sketched in (43):

(43) a. Topic {topic}
    b. Q {Wh, Neg, measure, focus, ... }
    c. Mod(ifier) {evaluative, epistemic, Neg, frequentative, celerative ... }

(44)
```
            Q
           /|
          / | M
         /  |  A [φ'sCase]
        /   |   
       /    |    
      /     |     
     /      |      
    /       |       
   /        |        
  Specific Q  θ
     /          
    /           
   /            
  θ_e
```

Summarizing, in these pages I reviewed two types of approach (derivational Vs. representational), highlighting the differences that are most relevant from a computational perspective. I pointed out that two crucial improvements on the standard Minimalist model (moreover consistent with this framework) are Phillips’ left-to-right incremental processing idea and the cartographic assumption about specific functional features realized in distinct projections and grouped in macro-classes for the purposes of locality constraints.

The next pages will evaluate the state of the art with respect to the implementation of these ideas within realistic computational/formal models.

2.3 Computational Models

Linguistic frameworks presented in §2.2 mainly dealt with descriptive/explanatory adequacy, that is, relevant linguistic phenomena are captured using devices that are
at least abstractly plausible. Implementing these intuitions within a computational model is unfortunately not readily possible. As I mentioned in the introduction, this is because:

- theories are not precisely formalized, then many “technical” aspects are left underspecified; this turns out to be an heavy burden on the shoulders of the computational linguists, since the choices they have to make to fill these blanks, often have unknown empirical consequences; moreover it is not always clear how to encode transparently linguistic intuitions even if they seem (intuitively) very precise;
- once formalized, theories turns out to be unsound, that is, principles/rules are contradictory or incompatible then the system become inconsistent;
- once formalized, models are impossible to be implemented since too complex, that is, requirements in terms of space and time are unsustainable (as discussed in §1.4).

In order to frame these problems, here we will go through some models implementation/formalization that attained a significant level of transparency with respect to some relevant linguistic assumptions presented in §2.2. This would allow us to highlight where difficulties reside and why.

We should notice that, because of concreteness, many linguistic problems are often avoided or left underspecified in order to implement more light, manageable and tractable computational models.

2.3.1 Principle-based Parsing

Early 90’s are characterized by the attempt of building linguistic resources deeply inspired by prominent linguistic approach such as Principle and Parameters (mainly expressed within the Government and Binding framework, §2.2.1). An interesting research thread within this context has been classified as “Principle-based parsing” approach (Berwick et al.1991).
One important example of this trend is Fong’s book (Fong 1991). In his work, he addresses many important issues about the difficulty of approaching the Principle and Parameters theory from a computational perspective, however providing important insights and a significant empirical coverage with his model implementation.

Notably he succeeded in:

- defining any principle using a high level language specification (every principle has been formalized using a subset of the first order logic, Horn clause logic, following Prolog specifications);

- identifying important inefficiency issues such as principles ordering, the logical inconsistency of part of the theory, the underspecification of many linguistic intuitions and the low efficiency of principle-based systems.

The final implementation of the system comprehends 25 principles plus two macro, used to expand automatically the principles as computational routines. As we noted in §2.2.1 the principles behavior, within GB approach, can be distinguished in two classes: generator (e.g. free indexation, move $\alpha$) and filters (e.g. case filter); the predicted problem was that if filters do not constrain as soon as possible generators output, the number of hypothetical structures to be evaluated grows dangerously (even though any order would produce the same correct structural description, if any). Fong shows that this concern is quite realistic and that, apparently, there is not any a priori best ordering to be used to speed-up the parser in any context. In fact, depending on the sentence to be processed, principle ordering can be finely tuned in order to generate the minimum problem space to be explored. Fong uses “cheap” cues within the sentence in order to propose a pre-ordering before start parsing (for instance pronouns suggest that the application of binding principles filters should happen as soon as possible).
Once ordered, principles and grammar are compiled in L(eft-to-right)R(ightmost) parsing tables\textsuperscript{59}. The architecture used is in fact a slight modification of a LR(1) parser, consisting of a Finite State Automaton (using two tables, a transition table and an action table) and three record stacks (one to register the state of the FSA, one to store the partially built structural description, one to encode other contextual information).

An efficiency problem with this structure has been noted in Merlo 1996: once modules have a lot to communicate among them (that is, they are not informationally encapsulated in Fodor’s 1983 terms), a fully modular system is inefficient; this inefficiency can be somewhat overcome using a pre-compilation of part of the grammar (Merlo points out an interesting principled way to distinguish which part of the grammar to compile off-line, e.g. phrase structure information such as X’-theory, and which one do not compile, e.g. empty traces position assumptions).

Without exploring the details of this operation, the intuition behind this assumption should be clear: despite of the premises, the relation between grammar and parser can hardly be attained within this model; linguistic principles, such as the ones analyzed in Fong, are not realistically suitable for a parsing model (in the precise sense proposed in the introduction): for instance an unbounded look-ahead, required to retrieve long distance relations, can produce a perfect parse, at the first attempt, even for garden path sentences, but this is not a “cognitively plausible” behavior (§2.2.3).

2.3.2 Minimalist formalization

Solving efficiency problems (both cognitive and computational) was one of the main reasons for moving from GB to Minimalism. Unfortunately, implementing a

\textsuperscript{59} Aho et al. 1986.
computational system inspired to the Minimalist Program was not an easy task. The dynamic nature of the inquiry produced many (often contradictory) ideas (e.g. *earliness*, Pesetsky 1989, Chomsky 2000 Vs. *procrastinate*, Chomsky 1995). Actually, this extremely active environment is still evolving and it is hard to freeze it in a consistent computational theory, moreover many crucial points are dramatically still underspecified. Within this perspective, Stabler’s (1997) work is particularly valuable since it provides the cleanest (and coherent) formalization of the Minimalist Program (the version proposed in Chomsky 1995): following his intuitions, a minimalist grammar can be defined as a 4-tuple \{V, Cat, Lex, F\} such that:

\[(53)\] **Minimalist Grammar** (MG, definition from Stabler 1997)

\[V\] is a finite set of non-syntactic features, \((P \cup I)\) where

- \(P\) are phonetic features and \(I\) are semantic ones;

\[Cat\] is a finite set of syntactic features,

\[Cat = (base \cup select \cup licensors \cup licensees)\]

- \(base\) are standard categories \{tense, verb, noun ...\};
- \(select\) specify 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 noun phrase;
- \(licensees\) force phrasal movement \{-wh, -case ...\}, \(-x\) triggers covert movement, while \(-X\) triggers overt movement;
- \(licensors\) features satisfy \(licensee\) requirements \{+wh, +case ...\}

\[Lex\] is a finite set of expressions built from \(V\) and \(Cat\) (the lexicon);
$F$ is a set of two partial functions from tuples of expressions to expressions

\{merge, move\};

The language defined by such a grammar would be the closure of the lexicon ($Lex$) under the structure building operations ($F$). (54) is a simple example of MG able to deal with simple $wh$-movements\(^{60}\):

(54) MG example

\[
\begin{align*}
V &= \{'what/', 'did/', 'you/', 'see/'\}, \\
I &= \{'what\}, \{'did\}, \{'you\}, \{'see\}, \\
Cat &= \{'D, N, V, T, C\}, select = \{'=D, =N, =V, =T, =C\}, \\
&\text{licensors }\{+wh\}, \text{ licensees }\{-wh\} \\
Lex &= \{'wh Dwhat\}, \{'-V Tdid\}, \{'D you\}, \{'D D V see\}, \{'T +wh C ϕ\} \\
F &= \{'merge, move\} \text{ such that:} \\
merge \(X, Y\) &= \text{a function taking two adjacent subtrees } X \text{ and } Y, \\
\text{outputting an unified structure } Z \text{ of the form } [_X X Y] \text{ if and only if } X \text{ has as first selecting feature } (=f, =F, =F) \text{ and } Y \text{ has the needed selected feature } F \text{ as the first feature of the base set} \\
move \(X, Y\) &= \text{if a function taking two subtrees } [_X X] \text{ and } [_X Y] \text{ such that } [[_X X] [W [_X Y]]] \text{ (where } W \text{ can be any possible subtree, even null, but without any selecting/selector feature } g \text{ in it) and produces } Z \text{ of the form } [[_X Y X] [W [t_f]]] \\
\end{align*}
\]

\(^{60}\) 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). Also Merge can be simplified avoiding directional selection, as Stabler suggests, by assuming that complex trees (i.e. results of other merge operations) always merge to the right. This creates a right branching structure, at least for simple tree-structures (i.e. structures in which the specifier is not complex, in the sense just discussed).
Following Chomsky (§2.2.2), a derivation proceeds bottom to top and licensees trigger movement as shown in (55):

(55) 1. merge ([D=V see], [wh D see what]) \(\rightarrow\) [D=V see [wh see what]]

2. merge ([D you], [D=V see [wh what]]) \(\rightarrow\) [v [you] [v see [wh what]]]

3. merge ([V T did], [v [you] [v see [wh what]]]) \(\rightarrow\)

   [v did [v [you] [v see [wh what]]]]

4. merge ([T+wh C ∅], [T did [v [you] [v see [wh what]]]]) \(\rightarrow\)

   [wh C ∅ [T did [v [you] [v see [wh what]]]]]

5. move ([wh C ∅ [T did [v [you] [v see [wh what]]]]) \(\rightarrow\)

   [C [what] C [T did [v [you] [v see [wh 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 2001). However, it is difficult to draw any cognitive/computational (i.e. realistic) conclusion from these results, since, for instance, the recognizer is based on a deductive parsing perspective (Shieber et al. 1995) that is not a cognitively motivated procedure and the equivalence results are based on a weak equivalence: namely, other formalisms/derivations can produce the very same set of strings MGs will produce, but either they fail to associate the same structures to these strings or they encode lexical items, features and structure 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

---

61 This is a very simplified version of derivation, to be taken only as example. It would be clearly possible including subject movement too, but this would have been required extra steps in the derivation.
formalization is to make clear what the computational/cognitive implications are either in parsing or in generation.

In this respect, very little can be said about $V$ and $Lex$, largely underspecified within the Minimalist Program: Stabler’s formalism on these points makes the simplest possible assumptions, worth to be kept the way he defined them.

Some important problems however arise with this formalization: the organization of $Cat$ in four subclasses of features, for instance, is not completely satisfactory given the cartographic analysis provided in §2.2.4: $base$ and $select$, the sets of standard categories, collapse together $functional$ and $lexical$ features ($Tense, V, D, N...$), even though there are strong empirical reasons to believe that this distinction is both theoretically (Belletti, Rizzi 1996) and cognitively justified (Leonard 1998, see §3.3 on this point). Furthermore, such a simple categorization would not be able to predict the correct locality constraints on movement: in these terms, locality would be just a matter of features identity, although it has been shown that this cannot be empirically adequate (§2.2.4).

Another problem is related to the fact that this formalization is based on early version of the Minimalist Program (Chomsky 1995) then, for instance, the notion of $phase$ (§2.2.2) is completely ignored. This allows the grammar to retrieve/produce arbitrarily distant relations, clearly not a welcome generalization (§2.2.2, §2.2.4).

2.3.3 Minimalist Parsing

Efforts in implementing the minimalist framework have been made not only on the formalization side, but also on the parsing perspective: Fong (2004), again, explores recent minimalist intuitions (§2.2.2, §2.2.3) trying to build a parsing model that makes crucial use of technologies such as the $probe-goal$ driven theory of movement and the idea of $derivation by phase$. His work is interesting since it implicitly shows how the Minimalist framework is not suitable for parsing using a bottom-to-top perspective. In fact, having a left-to-right parsing model (somehow similar to
Phillips’ model presented in §2.2.3) that aims to build structure incrementally, it is hard to make direct use of Merge and Move; for instance assuming that a complement is merged with its head before the specifier, from a parsing perspective, the left-to-right flow of information bears to the attention of the parser the specifier before the head or the complement (even in head-final languages like Japanese):

(56) a. Mary_{spec} read_{head} a book_{complement}
   b. Mary-ga_{spec} hon-o_{complement} yonda_{head} (Japanese)
   Mary-nom book-acc read
   ‘Mary read a book’

This would cause the parser to wait before merging the specifier with the head until the head-complement cluster is built. Having a pending item, the specifier, not yet integrated in the structure when the next words are parsed would cause backtracking.

Fong’s incremental parser partially solves this problem assuming that the processing proceeds assembling elementary trees in a way that is reminiscent of the Tree-Adjoining Grammars (TAGs, Joshi 1985):

(57) parsing algorithm
   step 1 – given a category C, pick an elementary tree headed by C then start consuming input tokens:
   step 2 – fill in the specifier (if one is predicted from the elementary tree selected)
   step 3 – fill in the head
   step 4 – fill in the complement(s) by recursively calling the parsing algorithm with C’ where C has the lexical property of selecting C’

Once the right category C is selected, no choice points should be encountered during the parsing and the integration/rejection of the input tokens will be driven by simple top-down expectations (nothing but elementary trees are considered). As we saw
before, Chomsky’s model avoids ambiguities/non-determinism, assuming a numeration process during which the elements that enter the computation are pre-selected and arranged in a Lexical Array (LA). Unfortunately, no LA is available for on-line parsing since any element that has to be integrated is revealed for the first time only when read from the input (§2.2.3). Then the critical point of the simple model sketched in (57) is located in the first step, namely the procedure of picking up the “right” elementary tree. Before addressing this issue, let us explore the lexicon and the structure building devices that Fong includes in his parser. First, but this is a standard habit, Fong’s lexicon remains underspecified with respect to phonological and semantic features, mainly encoding syntactic properties in a way that can be rephrased as follows (note the similarity with Stabler’s Cat set of features, §2.3.2):

\begin{align*}
(58) \text{Lexicon} \\
LI = \{ \text{Cat, P, uF, F} \} \text{ where} \\
\text{Cat (formal categories)} = \{ V_{\text{transitive}}, V_{\text{unaccusative}} \ldots, C_{\text{nh}} \ldots, N_{\text{referential}}, N_{\text{expletive}} \} \\
\text{P (properties/functions)} = \{ \text{select}(x), \text{value}(\text{case}(k)) \ldots, \text{spec}(\text{select}(x)) \} \\
\text{uF (uninterpretable features)} = \{ \phi \text{-features, other} \} \text{ where} \\
\phi \text{-features} = \{ \text{person, number, gender } \ldots \}, \text{other} = \{ \text{EPP } \ldots \} \\
\text{F (interpretable features)} = \{ \text{person, number } \ldots \}
\end{align*}

This formalization can be represented by a table like the following one:

---

\(^{62}\) Unfortunately Fong does not provide any effective cue to solve this issue.
(59) **Lexicon fragment** (Fong 2004)

<table>
<thead>
<tr>
<th>Main category</th>
<th>Cat</th>
<th>( P )</th>
<th>( uF )</th>
<th>( F )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>( \phi )-features</td>
<td>Others</td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>( v^{\text{trans}} )</td>
<td>select(V)</td>
<td>per(P)</td>
<td>epp</td>
</tr>
<tr>
<td></td>
<td></td>
<td>spec(select(N))</td>
<td>num(N)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>value(case(acc))</td>
<td>gen(G)</td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>( v_{\text{unacc}} )</td>
<td>select(V)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( v_{\text{unerg}} )</td>
<td>select(V)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( v_{\text{raising}} )</td>
<td>select(N)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>( v_{\text{trans}, \text{unacc}} )</td>
<td>select(N)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( v_{\text{unerg}} )</td>
<td>select(V)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( v_{\text{raising}} )</td>
<td>select(T-( \phi ))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>( T )</td>
<td>select(v)</td>
<td>per(P)</td>
<td>epp</td>
</tr>
<tr>
<td></td>
<td></td>
<td>value(case(nom))</td>
<td>num(N)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>gen(G)</td>
<td></td>
</tr>
<tr>
<td>T-( \phi ) (infinitive)</td>
<td>( T )</td>
<td>select(v)</td>
<td></td>
<td>epp</td>
</tr>
<tr>
<td>C</td>
<td>( C )</td>
<td>select(T)</td>
<td>epp</td>
<td>wh</td>
</tr>
<tr>
<td></td>
<td>( C_{\text{wh}} )</td>
<td>select(T)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>( N_{\text{referential}} )</td>
<td></td>
<td>case(_)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( N_{\text{wh}} )</td>
<td></td>
<td>wh</td>
<td>case(_)</td>
</tr>
<tr>
<td></td>
<td>( N_{\text{expletive}} )</td>
<td></td>
<td>per(P)</td>
<td></td>
</tr>
</tbody>
</table>

Where:
- \( \text{case(_,)} \) instantiates an open slot (namely a feature) for case values;
- \( \text{select}(x) \) is a function that selects an element headed by \( x \);
spec(select(x)) is a function that selects, in the specifier position, an element headed by x;
value(case(k)) assigns k case to an open slot;
epp is an uninterpretable feature able to trigger movement; it legitimates a specifier position as landing for movement;

Then elementary trees can be drawn from the lexicon, projecting the basic functional/lexical heads following their selectional properties:

(60) *elementary trees* (Fong 2004)

```
a. C
   |
 b. T
   |
   T
 c. T
   T/T-p
 d. N

e. v
   |
 v_unacc
 f. v
 g. V
 V_trans/unacc/raising
 h. V
 V_unerg
   v*
   v_{unerg}
```

Fong suggests that, in order to keep the parsing algorithm (locally) deterministic, we should underspecify the tree structure when more than one elementary tree is structurally compatible: for instance, once an argumental NP is found, that is compatible with the specifier position both of a transitive and of an unergative v elementary tree, the algorithm simply projects the head (and the rest of the structure, if any) shared by both, keeping the rest of the non-matching structure underspecified; in this case, we could project the V complement, predicted both by
the elementary tree headed by $T$ and by the one headed by $T_{φ}$, but not the property value($\text{case(acc)}$), present only in $T$.

On the other hand, this model incorporates devices able to deal with Long Distance Dependencies: a Move Box and a Probe Box. The Move Box is somehow reminiscent of the HOLD register of Augmented Transition Networks (ATN, Woods 1970): potential trace fillers are put in a sort of short-term memory buffer, the Move Box, in a principled manner and retrieved when needed to fill the relevant empty positions in the sentence. Fong provides a precise algorithm that controls the behavior of this devices:

(61) Move Box (MB)

i. (initial content) at the beginning of any parse, initialize MB as empty;
ii. (fill condition) whenever an open position of an elementary tree is filled from the input, copy the filling element from the input to the MB;
iii. (preference rule) whenever an open position of an elementary tree can be filled by the element in the MB, fill it before looking at the input,
iv. if step iii is successful, remove the element from the MB;
v. (conditions on iv, empty condition for expletives) if in MB there is an expletive, then, after filling a position with it, optionally remove it from the move box.

Using MB requires a refinement of the parsing algorithm given in (57) that can be done simply assuming that fillers can come both from the input and from the move box (the preference rule given in (61) avoids non-determinism that would be caused by the option of choosing a filler either from the input or from the MB).

This device is sufficient to deal with many simple movement phenomena:
John saw Mary (assume saw → past + see)

<table>
<thead>
<tr>
<th>step</th>
<th>partial phrase structure built</th>
<th>MB</th>
<th>input</th>
<th>operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>c [ ]</td>
<td>ø</td>
<td>J. saw M.</td>
<td>c:select(T)</td>
</tr>
<tr>
<td>2</td>
<td>[c [T [T T [ ]]]]</td>
<td>ø</td>
<td>J. saw M.</td>
<td>fill spec-T</td>
</tr>
<tr>
<td>3</td>
<td>[c [T J [T T [ ]]]]</td>
<td>J.</td>
<td>saw M.</td>
<td>fill head-T</td>
</tr>
<tr>
<td>4</td>
<td>[c [T J [T past [ ]]]]</td>
<td>J.</td>
<td>see M.</td>
<td>T:select(v)</td>
</tr>
<tr>
<td>5</td>
<td>[c [T J [T past [v [v [v v see ]]]]]]</td>
<td>J.</td>
<td>see M.</td>
<td>fill spec-v with t from m.-box</td>
</tr>
<tr>
<td>5'</td>
<td></td>
<td>ø</td>
<td>see M.</td>
<td>free m.-box</td>
</tr>
<tr>
<td>6</td>
<td>[c [T J [T past [v t [v v* [v v see ]]]]]]</td>
<td>ø</td>
<td>see M.</td>
<td>v:select(V)</td>
</tr>
<tr>
<td>7</td>
<td>[c [T J [T past [v t [v v* [V V see ]]]]]]</td>
<td>ø</td>
<td>see M.</td>
<td>fill head-V</td>
</tr>
<tr>
<td>8</td>
<td>[c [T J [T past [v t [v v* [V see ]]]]]]</td>
<td>ø</td>
<td>M.</td>
<td>V:select(N)</td>
</tr>
<tr>
<td>9</td>
<td>[c [T J [T past [v t [v v* [V see N ]]]]]]</td>
<td>ø</td>
<td>M.</td>
<td>fill head-N</td>
</tr>
<tr>
<td>10</td>
<td>[c [T J [T past [v t [v v* [V see Mary ]]]]]]</td>
<td>M.</td>
<td>ø</td>
<td>parse found!</td>
</tr>
</tbody>
</table>

From this example, it should be clear that the fill condition is actually applied only to argumental elements in order to deal with A-movements. In fact, the current implementation is able to deal, at best, with the simplest locality conditions (features identity) leaving unaccounted any Relativized Minimality effect (Rizzi 1990, §2.2.4).

Moreover, there is an extra price to be paid to capture cyclic movement: i.e. the introduction of a choice option (namely a non-deterministic device) any time we fill a trace; for instance, with an expletive, the parser should choose whether to keep it in the MB, predicting a cyclic movement, or to remove it from the MB (remember that keeping it until the end of the sentence would cause the wrong behavior of inserting a “there-trace” instead of using “prizes” as shown in (63).b):

(63) a. there are supposed tthere to be likely tthere to be prizes awarded
       (free the move-box before “prizes”)

   b. there are supposed tthere to be likely tthere *prizes
       (keep the expletive till the end of the sentence)
This non-determinism (as discussed also by Fong) is present in any instance of successive cyclic movement:

(64) Several prizes are likely to be awarded *r?.

After dropping the trace with several prizes, the MB is emptied because of (61).d. So we would need a more efficient way of controlling the lifespan of the elements in the MB. Moreover, it is possible to have multiple phrasal movements in the same sentence:

(65) Who was a book given to who

Fong deals with this fact postulating nested move boxes; as the HOLD register in the ATNs, this memory has a LIFO structure (Last In, First Out): the parser can only see the last inserted element, namely the last created move box is the first available filler. But this would not predict the correct result, for instance, in the following case, namely when movements overlap:

(66) Who did Bill see who

Fong assumes that overlapping should require various MBs and crucially a more powerful machinery than nesting, given that anytime we have multiple MBs full of potential fillers we would have other choice points, namely a non-deterministic processing.

Last device Fong proposes is a Probe Box for encoding the essential agreement relation in the parser. As introduced in §2.2.2, Minimalist Grammars require that an agreement relation hold when a probe merges with a goal.

Once agreement holds, the goal values the unvalued features on the active probe. This is the proposed algorithm ($p = \text{probe}, g = \text{goal}, f = \text{feature}, \alpha, \beta = \text{generic syntactic objects}$):
(67) *Agree*(p, g) if
   i. Match(p, g) holds, then
   ii. Value(p, g) for matching features and
   iii. Value(p, g) for property value(f)

(68) *Value*(α, β) holds if
   i. (Unification) Unify matching φ-features values of α and β
   ii. (Assignment) If α has the property value(f), then f in β receives its value from α

The Probe Box has to be intended as a special slot of short-term memory where active elements (namely elements bearing uninterpretable features) are made available for checking purposes. The algorithm that controls this tool is similar to the one used for the move box:

(69) *Probe Box* (PB)
   i. (initial content) at the beginning of any parse, initialize PB as empty;
   ii. (fill condition) whenever an head position of an elementary tree is filled from the input with an element bearing one or more uninterpretable features, copy this element in PB;
   iii. (unicity condition) only one probe is allowed to stay in PB: new probes found in the input replace old ones.

At this point, the parsing algorithm given in (57) has to be redefined as follows:

(57)’ *parsing algorithm* (refined)
   step 1 – given a category C, pick an elementary tree headed by C using elements from the move box or from the input:
   step 2 – fill in the specifier s (if one is predicted from the elementary tree selected)
   step 3 – run *Agree*(p, s) if p and s are non-empty
   step 4 – fill in the head h
step 5 – run $Agree(p, h)$ for $\phi$-incomplete $h$ and $p$ non-empty

step 6 – copy $h$ to PB as $p$ if $h$ is a probe

step 7 – fill in the complement(s) by recursively calling the parsing algorithm with $C'$ where $C$ has the lexical property of selecting $C'$

We can now see how the probe box behaves in the simple case presented in (62), reporting below the new operations required in order to deal with agreement:

$\phi$-incomplete $h$ and $p$ non-empty $Agree(p, h)$

PB

We can now see how the probe box behaves in the simple case presented in (62), reporting below the new operations required in order to deal with agreement:

(62) John saw Mary (assume saw $\rightarrow$ past + see)

As Fong points out, we attain at least two good services from this device:

- **Phase Impenetrability Condition**: a probe cannot penetrate into the domain of a lower probe since it is ousted by this lower probe from PB;

- **Valuation of $\phi$-incomplete probes**: unifying probes with heads as in (68).i allows the unvalued $\phi$-features of these heads to receive features; then if these heads become probes (this is the case of the defective $T_\phi$) they will bear a complete featural make up so as to satisfy further agreement operations as for the following sentence:

(70) we expect there to arrive a man

$\phi$-incomplete $h$ and $p$ non-empty $Agree(p, h)$

PB

As Fong points out, we attain at least two good services from this device:

- **Phase Impenetrability Condition**: a probe cannot penetrate into the domain of a lower probe since it is ousted by this lower probe from PB;

- **Valuation of $\phi$-incomplete probes**: unifying probes with heads as in (68).i allows the unvalued $\phi$-features of these heads to receive features; then if these heads become probes (this is the case of the defective $T_\phi$) they will bear a complete featural make up so as to satisfy further agreement operations as for the following sentence:

(70) we expect there to arrive a man

$\phi$-incomplete $h$ and $p$ non-empty $Agree(p, h)$

PB

As Fong points out, we attain at least two good services from this device:

- **Phase Impenetrability Condition**: a probe cannot penetrate into the domain of a lower probe since it is ousted by this lower probe from PB;

- **Valuation of $\phi$-incomplete probes**: unifying probes with heads as in (68).i allows the unvalued $\phi$-features of these heads to receive features; then if these heads become probes (this is the case of the defective $T_\phi$) they will bear a complete featural make up so as to satisfy further agreement operations as for the following sentence:

(70) we expect there to arrive a man

$\phi$-incomplete $h$ and $p$ non-empty $Agree(p, h)$

PB

As Fong points out, we attain at least two good services from this device:

- **Phase Impenetrability Condition**: a probe cannot penetrate into the domain of a lower probe since it is ousted by this lower probe from PB;

- **Valuation of $\phi$-incomplete probes**: unifying probes with heads as in (68).i allows the unvalued $\phi$-features of these heads to receive features; then if these heads become probes (this is the case of the defective $T_\phi$) they will bear a complete featural make up so as to satisfy further agreement operations as for the following sentence:

(70) we expect there to arrive a man

$\phi$-incomplete $h$ and $p$ non-empty $Agree(p, h)$

PB

As Fong points out, we attain at least two good services from this device:

- **Phase Impenetrability Condition**: a probe cannot penetrate into the domain of a lower probe since it is ousted by this lower probe from PB;

- **Valuation of $\phi$-incomplete probes**: unifying probes with heads as in (68).i allows the unvalued $\phi$-features of these heads to receive features; then if these heads become probes (this is the case of the defective $T_\phi$) they will bear a complete featural make up so as to satisfy further agreement operations as for the following sentence:

(70) we expect there to arrive a man

$\phi$-incomplete $h$ and $p$ non-empty $Agree(p, h)$

PB
This justifies the permanence of a probe in PB also after goal valuing (namely, when it becomes inactive): even if it cannot trigger further valuing procedures (e.g. \textit{value}(\textit{case}(k))), it can still unify its features with the following heads as for \(T_{\phi}\).

Summarizing, this paring strategy addresses some important issues:

i. it catches the ungrammaticality of the sentence as soon as an unpredicted element enters the computation;

ii. it encodes movement (and residual properties of the chains) using a memory buffer, the Move Box;

iii. it correctly predicts (some) structural case assignment, approximating, moreover, the strong phase boundaries using another memory buffer, the Probe Box;

iv. it is incremental, namely a partial parse is always available at any stage of the processing (this suggests us that some relevant psycholinguistic phenomena could be captured);

v. it exhibits a locally deterministic computation, namely no choice points are present, once chosen the correct category (then the correct elementary tree);

vi. it is an on-line procedure, that is, the lifespan of an element is restricted to its integration point in the structure, then, it is immediately forgotten unless saved in the Move/Probe box.

Despite these interesting results, the parsing model proposed by Fong is still imperfect under many points of view:

i. it is not clear how to pre-select the correct category or how to backtrack from a wrong choice;

ii. neither locality conditions nor cyclic movement is easily catchable by using the Move Box as described in the algorithm;
iii. the current implementation of two memory buffers within the same model seems to be inefficient from several points of view:

a. a redundancy seems to be present implementing both a Move Box and a Probe Box, since the first one, in fact, should be able to trigger agreement per se (any A'-movement is toward a criterial functional position, Rizzi 2004, then, by definition of functional position, cf. Fseq in §1.1.5, any moved element should agree somehow with the landing site head);

b. populating the Move Box is too easy and quite expensive in terms of memory resources (following the fill condition proposed in (61).ii, any element is potentially copied in the move box; many times this turns out to be unnecessary, as in (62), step 10, where “Mary” should be vacuously moved to the Move Box);

c. the preference rule for the Move Box sometimes seems to predict the wrong expectation (e.g. “who did Mary [read the book of twho]?” Vs. “who did Mary [read twho] the book of?” as wrongly predicted by the algorithm).

These inefficiencies will be tackled in the next chapter.
With the goal of integrating what has been discussed so far, I want to start this chapter by critically analyzing the notion of Structural Description (SD): here I propose (§3.1.1) that a SD can be given simply in terms of immediate relations among elements (a simplified version of the notion of dominance and precedence presented in §1.3) this will turn out to be appropriate both from a parsing and from a generation perspective (§3.1.2) and will allow us to frame, in a precise way, two problematic aspects of phrase structure: ambiguity (§3.2.1) and non-local dependencies (§3.2.2).

The third and fourth part of this chapter will deal with Structure Building Operations: Merge (§3.3.2) and Move (§3.3.4) will be redefined from a top-down perspective, in order to solve some empirical problems (§3.3.1, §3.3.3) emerging both at the theoretical (§2.2) and at the computational (§2.3) level. Finally, the notion of Phase will be formalized (§3.4.3); this will permit us to attain a satisfactory complexity reduction both in parsing and in generation with respect to ambiguity (§3.4.1) and non-local dependencies (§3.4.2).

3.1 Aspects of the Grammar to be formalized

In order to identify the core characteristics that a minimal specification of the grammar should consider so to map features structures both at the sensory-motor interface and at the conceptual-intentional one, we should define:

- the relevant relations among features to be specified within an appropriate Structural Description, readable from any performance perspective (§3.1.1; this refines the discussion in §1.3);
- the exact specification of the performance tasks, namely parsing and generation, which should access to the grammar (§3.1.2);
3.1.1 The bidimensional nature of the grammar

So far, considerations about the relevant relationships among the grammatical entities to be included in a structural description, leaded us to think of 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 sequences of phonological features) while dominance expresses the constituency/dependency relations among pronounced and other implied (abstract) elements (semantic and other abstract features like phrase identifiers). These two kinds of information have been usually encoded within tree-like structures such as the following one:

(1)

A language, as we defined it so far, is an infinite set of expressions each associated with at least one grammatical structural description, which productively restricts the theoretically possible precedence/dominance relations that can co-occur within the same sentence.

These properties are usually encoded in Structural Descriptions defined as 5-tuples such that (Stabler 1997):
(2) \( SD_{standard} = \{I, P, D, V, A\} \) (definition)
where
- \( I \) is a finite set of identifiers (e.g., \{the, dog, is, ..., DP, D', D°, N°\})
- \( P \) 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 \( I \)

This formalization allows us to build tree structures like the ones presented in (1) or in (3).a (which is equivalent to the labeled bracketing (3).b; §1.1.4):

(3) a. \[
B'' \\
A
\]

b. \[
[\[ A \] B C ]
\]

It seems empirically relevant (e.g. for cliticization phenomena, Zwicky (1977)), given an ordered set of elements \(<A, B, C>\), to distinguish the relation between \( A \) and \( B \) (immediate precedence, that is, no elements intervene between \( A \) and \( B \)) from the relation between \( A \) and \( C \) (standard precedence). The same seems to be true also
in terms of dominance: given a dominance structure such as $[\verb [Arg1 [Arg2]]]$ (that is, the verb dominates Arg$_1$, Arg$_1$ dominates Arg$_2$, and the verb dominates Arg$_2$) a specific verb assigns a specific thematic role to the immediately dominated argument (Arg$_1$) which is not compatible with the thematic argument to be assigned to another dominated position (e.g. Arg$_2$ in a sentence like “Mary $[\verb bought [Arg1 the picture [Arg2 of John]]]$” cannot receive a role by the verb).

Because of the empirical concerns, let me redefine the first three elements in a way that precedence and dominance only hold between immediately adjacent elements:

(4) $SD_{\text{revised}} = \{I, P, D\}$ (definition)

such that

$I$ is a finite set of identifiers (only lexical items such as \{the, dog, is, black...\}, following the inclusiveness condition, §2.2.2, are part of this set)

$P$ is a finite set of immediate precedence relations (different from the classic total strict order, this relation is only defined between two adjacent items) example ($<A, B>$ means $A$ immediately precedes $B$):

$P = \{<A, B>, <B, C>, <C, D>\}$

(simplified in $<A, B, C>$)

$D$ is a finite set of immediate dominance relations (different from the classic dominance relation, this one is a partial, binary, intransitive and asymmetric relation)

example ($A < B$, means $A$ dominates $B$):

$D = \{A < B, B < C, C < D\}$

(equivalent to $[A [B [C D]]]$)
These restricted versions of $P$ (recendence) and $D$ (ominance), defined only between adjacent elements, will be useful to encode in an extremely transparent way some significant linguistic relations (e.g. Merge). In the next pages I will refer to precedence and dominance as immediate relations, unless specified otherwise. Notice that the arboreal representations given in (5).a and (5).b are equivalent (adapting Stabler’s 1997 convention, $<$ means that the symbol on the edge of the arrow immediately dominates the symbol at the base of the arrow):

\[
\begin{align*}
(5) & \quad \text{a} & \quad \text{b}. \\
\ \ & \quad A & \quad B & \quad A & \quad B \\
\ & \quad A & \quad B & \quad A & \quad B \\
\ & \quad B & \quad C & \quad B & \quad C \\
\ & \quad < & \quad > & \quad < & \quad > \\
\end{align*}
\]

3.1.2 Performance tasks

Recall that the speaker’s competence is the intensional procedure that characterizes an infinite set of sentences and that, formally speaking, this competence is represented by a grammar. The grammar includes, by standard hypothesis, at least a specification of a lexicon (a finite set of words built from an alphabet with associated specific features) plus some universal properties (usually encoded as principles) and language specific options (parameters) to derive the combinatorial potentiality of any natural language.

As we saw in §2.1 and in §2.3.1, from this perspective, the specification and ordering of Structure building operations is controversial: namely, it could be superfluous (or impossible) and even illegitimate (if we consider them as part of the parser and not as a part of the grammar) to specify some precise algorithm that

---

63 Let us assume by “standard hypothesis” the Principle and Parameter framework presented in Chomsky 1981 and quickly reviewed in §2.1.1.
recursively defines the procedure for assembling bigger and bigger meaningful units, starting from the lexicon and its properties. The minimalist framework, for instance, pays serious attention to this point, both providing interesting solutions (§2.2.2, §2.2.3, §2.3.2, §2.3.3) but also raising puzzling problems (§3.3). As I pointed out in the previous chapters, I will try to consider, in line with the Minimalist Program (§2.2.2), Phillips’ left-to-right model (§2.2.3) and Stabler’s formalization (§2.3.2), the Structure building operations as a part of the grammar specification. Then I will explore, in order to understand some core properties of these operations, two performance tasks from a formal perspective, eventually arguing in favor of a revision of the directionality of these operations within a minimal specification of the grammar.

The first task I will consider from this perspective is parsing: this means recovering a set of dominance relations from the precedence relations holding among a given set of lexical items, minimally in the form \{\pi, \sigma\} where \pi is a set of phonetic features and \sigma a set of semantic features. This problem can be stated more formally as follows:

\[(6) \text{ parsing problem (definition)}\]
\[
\text{given a grammar } G, \text{ a finite set of phonological features } \pi (\text{grouped by words}) \text{ and a precedence total order among them, find the relevant set of lexical items } Lex, \text{ compatible with } \pi \text{ and the set of dominance relations } D \text{ among } \sigma \text{ features associated to } \pi \text{ in } Lex, \text{ if possible, if not reject the input.}\]

The second task is “production”, which means generating a sequence of lexical items, compatible with a given set of dominance relations defined on a set of semantic features; more precisely:

\[(7) \text{ generation problem (definition)}\]
\[
\text{given a grammar } G, \text{ a finite set of semantic features } \sigma \text{ and a finite set of dominance relations } D \text{ among them, find the relevant set of lexical items}\]
Lex and the correct linearization among $\pi$ features associated to $\sigma$ in Lex, if possible, if not reject the input.

Let us now try to test these problems with some real linguistic data.

3.2 Two (computationally hard) linguistic phenomena: ambiguity and non-local dependencies

Two properties of any natural language are extremely hard to be captured, computationally speaking, even though they apparently do not present any problem for a native speaker: the first property is that language is massively ambiguous; the second one is that the linguistic elements seem to be related among them in a way that is not always self-evident looking at linear order in which they appear in the sentence.

While the second property, also known as Non-Local (or filler-gap, Fodor 1978) Dependency, has always been considered as irreducible core part of the grammar, the ambiguity issue has been often relegated to the domain of processing, that is, to the parsing task. A proper understanding of their relation to the grammatical knowledge requires first an empirically grounded precise formalization of these properties (§3.2.1, §3.2.2).

3.2.1 Ambiguity

Ambiguity arises any time an input can be analyzed in a non-deterministic way, that is, given a certain amount of information, the processor can make multiple choices that will lead to different outputs. For instance a word like “dog”, taken in isolation, can be either a noun (“the dog is in the garden”) or a verb (“did you dog me?”). Providing a bit more of context (e.g. some words preceding the target one) could help solving the problem. Thus ambiguity seems to be a property originating both from the input and from the algorithm (based on the grammar) that drive the computation.
Among the levels of ambiguity present in any natural language, we can distinguish at least three macro classes: *lexical*, *syntactic* and *semantic* ambiguities.

A *lexical ambiguity* is present when multiple categories, expressing a Part-of-Speech (PoS) can be associated to a single word in a sentence:

(8) a. I read the [*Noun book]*
    b. Did you [*Verb book*] the flight?

On the other hand, we have a *syntactic ambiguity* when more than one structural description for a given sentence is possible:

(9) a. Please, [*put [the block] [in the box] [on the table]*]
    a'. Please, [*put [[the block] in the box] [on the table]*]
    a". Please, [*put [[the block] in the box] [on the table]*]
    b. [[Red roses] and [cherries]]
    b'. [Red [[roses] and [cherries]]]

A third possibility is having multiple senses associated with the same word (same PoS and same \( \pi \) features), this is a case of *semantic ambiguity*:

(10) a. Tonight the moon is full (moon = the earth’s unique satellite)
    b. Jupiter has sixteen moons (moon = orbiting object)

Given a SD such as the one explored in (4), *lexical ambiguity* can be expressed in terms of abstract (categorial) features opposition, so that it is invisible to *precedence* and perhaps even to *dominance*. This is predictable, since these two relations were defined respectively only on \( \pi \) features (*precedence*) and on \( \sigma \) features (*dominance*).

On the other hand, syntactic ambiguity is identified by the existence of more than one *dominance* set of relations associated to the same *precedence* set of relations.

Finally, *semantic ambiguity*, is not marked by different *precedence/dominance* combinations, but rather by different sets of \( \sigma \) features associated to the same set of \( \pi \) features.
3.2.2 Typologies of Non-Local Dependencies

As we have seen in §1.3, among the relations we could define on elements within a SD, other than dominance and precedence, only few are linguistically relevant:

(11) Constituent formation

a constituent is formed when two adjacent elements are concatenated together (e.g. $B$ concatenates with $C$ forming the constituent $B'$ in (3))

(12) C(onsituent)-command  (Chomsky 2000)

$A$ $C$-commands $B$ iff $B$ is contained in the sister of $A$

(e.g. $A$ $C$-Commands $B$ in (3))

Notice that the relation in (11) is indeed perfectly equivalent to the dominance relation defined in (4). (12), on the other hand, can express a relation between arbitrarily distant elements and it is neither directly encoded within the precedence nor within the dominance definition, even though, in principle, it could be captured by adding transitivity to the dominance definition (§1.3).

So to speak, any relation between two non-adjacent elements in a SD can be considered as an instance of Non-Local Dependency.

More precisely, this configuration should hold:

(13)
where $X$ and $Z$ are identical in some fundamental properties (e.g. formal and/or $\sigma$ features) and $Y$ intervenes between $X$ and $Z$ without blocking the relation as shown below:

(14)a. $[X \text{ John }] [Y \text{ gave }] [Z \text{ his }]$, picture to Mary (pronominal binding)

b. $[X \text{ What }] [Y \text{ do you think }] [Z \text{ _ }]$,? (movement)

c. John $[X \text{ bought the book }] [Y \text{ and Mary did too }] [Z \text{ _ }]$ (ellipsis)

Remember that the blocking nature of the $Y$ category has been discussed in §2.2.4.

The identity relation between $X$ and $Z$ could be considered directional, in the sense that the structurally “higher” element provides the feature values for the lower (invisible/incomplete) one (this justifies the direction of the arrow). Notice that the nature of $X$, $Y$ and $Z$ can vary to some degree from a full phrase, as in (14).c, to a simple head, to a pronominal form (14).a, to an unpronounced element (14).b-c.

Given the formalization of SD in (4), we can define discontinuous constituency dependencies as follows:

(15)Non-Local Dependency (definition)

two non-empty elements enter a Non-Local Dependency (thus forming a discontinuous constituency relation) when an immediate dominance relation but no immediate precedence relation is defined between them.

For instance, given the grammar in (16), $A$ and $C$ are subject to a Non-Local Dependency since the dominance relation $C < A$ exists but neither $<A,C>$ nor $<C,A>$ is present:

(16)$I = \{ A, B, C \}$

$P = \{ <A, B>, <B, C> \}$

$D = \{ B < A, B < C, C < A \}$

---

64 Where non-empty means with at least some feature specified (either formal or $\sigma$ features).
This can be represented by the tree below:

(17)

The Non-Local Dependency in (16)-(17) is essentially an instance of movement (as discussed in §2.2), however it should be possible to extend this definition also to pronominal binding (along the lines of Kayne 2002) and to control (Hornstein 1999, Boeckx and Hornstein 2004), for instance. The relation between \( A \) and \( (A) \) is an “identity” relation between a moved element and its trace (again, the directionality of the arrow indicates that the highest element provides feature values for interpreting underspecified features of the lowest one).

Note that the information in (16) is ambiguous between the right-branching structure given in (17) and more complex structures such as the ones shown below:

(18)a. 

(18)b. 

139
We can rule out the unwanted representations, in a principled way, by posing extra constraints on the occurrence of Non-Local Dependency and/or on the general shape of the SD. The Linear Correspondence Axiom (Kayne 1994), for instance, would suggest a deterministic mapping between asymmetric C-command and precedence depending on the structural function of the items involved in the SD. In a similar vein, I will adopt the following principle:

(19) Linearization Principle (inspired by 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 of $A$

By now we do not need to discard (18).b since we might not have any need to distinguish it from (17): the linearization only applies $\pi$-features; since a trace is (often) $\pi$-features free, the linear order with respect to the item it enters in an immediate dominance relation, is undefined.

But we can discard (18).a under the relatively standard constraint (20):

(20) Constraint on movement (definition)

a moved element always C-commands its trace(s)

We are now ready to consider how these properties can be incorporated in a grammar that fulfills the (empirical) requirements of flexibility and token transparency as discussed in §2.1.

3.3 Empirical inadequacies: re-defining Merge and Move

The Minimalist framework raises interesting questions on the nature/power of the grammatical computational engine, the core part of the human linguistic competence apt to build phrase structures. However, to what extent the proposed abstract grammar formalization is usable both in parsing and in generation remains largely an unexplored topic. On the other hand it is plausible, from a cognitive perspective,
that the grammar we use to produce sentences is somehow used also to comprehend them (this is the flexibility property discussed in the introduction and in §2.1).

Here I want to show that the bottom-to-top orientation of the structure building operation Merge and Move is not suitable to meet these requirement and that we gain in explanatory adequacy deriving phrase structure top-down.

3.3.1 Cartography, Extended Projections and Phases
As we mentioned in §2.2.4, recent linguistic inquiry unveiled a rich architecture related to the presence of many, cross-linguistically ordered, functional projections (Cinque 1999, Starke 2009). A first problem here is related to how this order is derived. From a theoretical point of view, we could encode, within Stabler’s formalism (§2.3.2), such order by imposing the correct sequence by selection:

(21) Lexicon fragment and derived hierarchical structure:

\[
\text{Lex} = \{[\text{ordinal D}] [\text{cardinal ordinal}], [\text{… cardinal}], [\text{… N material} [x]]\}
\]

\[
\begin{array}{c}
D \\
\_ \text{ordinal} \\
\_ \text{cardinal} \\
\_ \text{cardinal} \_ \\
\_ \text{… material} \\
\_ \text{material} \_ \text{N}
\end{array}
\]

This is the standard (implicit) assumption most linguists make: selection creates hierarchies. This is however problematic from at least two points of view: first, the necessity of projecting a full extended projection is anti-economical, second, in case of optional selection, that would legitimate the absence of certain functional
projection, the number of items in the lexicon will increase exponentially. This is how it happens:

(22) lexicon fragment with selecting functional features:

\[
\text{lex} = \{=\text{Asp, Mood, probably}, =\text{V, Asp, suddenly}, [\text{N, Napoleon}, [\text{N, V, died}]]
\]

Given the lexicon fragment above, there are sentences that are generable using this lexicon fragment, like “probably suddenly Napoleon died”, but other that are not, like “*probably Napoleon died” “*probably Napoleon died suddenly”; in the first case the derivation fails while attempting at merging the V complex \([\text{V, Napoleon, died}]\) with \([=\text{Asp, Probably}]\); this second elements does not properly select V, hence the ungrammaticality of the sentence that should be, indeed, a good English sentence. In the second case, unless we assume (heavy) remnant movement, a lower adverbial cannot be selected neither by a distal adverbial or verb. Changing the \(=\text{Asp}\) requirement of \textit{probably} to \(=\text{V}\) would not solve the problem since, then, the first sentence “Probably suddenly Napoleon died” will not be generable anymore.

We can play around with selection features as we want, but the only possible solutions we have are two:

(23).a. introducing ambiguity in the lexicon;

b. assuming that functional features can be phonologically empty.

(23).a would produce the (factorial, i.e. \(n!\)) proliferation of functional heads in our lexicon \((=\text{Asp, Mood, probably}), (=\text{T, Mood, probably}), (=\text{V, Mood, probably})\ldots\). On the other hand, (23).b will be more economical, since every functional head will be simply duplicated \((=\text{x, Mood, probably}), (=\text{x, Mood, } \theta)\); but this would prevent the structure from being minimal since every tree would have the full functional projection in place, even though vacuously filled.

If this second solution is the only plausible way we have from a bottom-to-top, selection driven, derivational perspective, I think there is a more economical solution we can pursue inverting the orientation of the derivation from bottom-to-
top, to top-down: assuming a head-driven selection, I will consider $V$ and $N$ to be the only lexical categories to be included in the base/select subsets (§2.3.2). This way, since a verb, for instance, should select as a complement either another verb or a noun (the only features in the base/select class), these categories have to be maximal projections, then any functional projection (represented by a single licensor feature) above a lexical head should be dominated by this head. This entails the structure (24), namely some sort of Extended Projection along the lines of Grimshaw (1991):

$$\text{(24)}$$

In this book, I will assume that structures like (24) are phases in a sense very similar to the one discussed in Chomsky (1999) (§3.4.3):

$$\text{(25) Phase (definition)}$$

a phase is an extended projection of a lexical category

Phases (here I will discuss just $N(ominal)$ and $V(erbal)$ lexical phases) must be properly selected by standard select features on lexical items and they project

---

$^65$ Just for sake of simplicity, I will not include here lexical categories as (certain) adjectivals or adverbials. Nothing here hinges on the number of lexical head chosen.
thematic shells that qualifies as independent phases; when more select features are present on the same lexical head, I assume that these features will be projected, top-down, as sketched in (26)\(^{66}\):

\[
\text{(26)}
\]

```
(\text{\small V} = \text{\small DP}\text{\small = DP\text{\small call}}) \\
\text{\small V} \quad \text{\small VP} \\
\text{\small DP} \\
\text{\small V} \quad \text{\small DP} \\
\text{\small DP} \\
\text{\small V} \quad \text{\small DP} \\
\text{\small DP} \\
\text{\small V} \quad \text{\small DP} \\
\text{\small DP}
```

Such structure, created by select features expansion on the verbal head, is the same as Larson’s VP shells (Larson 1988).

We can now explore the consequences of these assumptions with respect to the directionality of the derivation and the definition of the structure building operations Merge and Move. This would lead us to avoid unwanted lexical explosions or vacuous functional hierarchy population, moreover better expressing relevant relativized minimality effects.

3.3.2 Merge reduces to Lexical Insertion

The simple idea of linking the merge operation to a simple, single-feature, selecting mechanism (=f) has been shown to be problematic in the previous paragraph. Introducing a distinction between functional and lexical features, however, does not solve the problem per se: there are functional elements that are optionally selected (for instance, adverbs) while others seem to be obligatory in some

\(^{66}\) For “backward compatibility”, I will refer to determined nominal phases ([\text{\small \text{DN}}]) as DPs, but, in fact, they should be considered extended nominal projections.
contexts/languages (e.g. determiners, tenses etc.). This asymmetry can be captured, within the Extended Projection assumption introduced in §3.3.1, if we re-define the selection mechanism: if selection would be able to express functional features, as well as lexical features, as requirements, it could control exactly the number of features to be present in certain structures. For instance, assuming that select features can express clusters of both licensor and base items, we can encode within a verb the “=[+D N]” selection requirement; this expresses the necessity of a noun lexical phrase (N) where the determiner licensor feature (+D) must be satisfied. The lexicon will look like this:

(27) \text{Lex} = \{ [+TV,+D\text{N}=+D\text{N}]\text{called}, [+D\text{N}\text{John}, [+D \text{the}, [N \text{police}, \ldots] \}

Notice that at this point, the bottom-to-top Merge (§2.2.2, §2.3.2) is emptied of its role, since elements like [+D \text{the}] and [N police] does not express any selection that allows them to combine.

On the other hand, assuming that the derivation proceeds Top-Down, if we would know that a selection requirement like =+[D N] must be satisfied, then we could proceed from left to right (as Phillips 1996 proposes) by lexicalizing first +D then N picking up the correct element from the lexicon (either [+D \text{the}], [N police], or [+D\text{N}\text{John}] that would lexicalize both features at once). Notice that if we assume that the lexicon is hierarchically structured and that features are higher nodes of this hierarchy, lexical inspection would be easier than expected: if we know that +D is a function features related to the lexical category N, our lexical inspection would first choose to pick up an item within the N-related partition of the lexicon, then choosing the +D subclass, and, eventually, other feature-related subclasses (like the ones originated by gender or number distinctions). This way, picking up the relevant

\footnote{As Stabler suggests, features in the lexical entries are ordered; the order here proposed is <+licensors*, lexical-head, =selections*>.}
lexical items would prevent the blind inspection of the whole lexicon, dispensing our grammar from necessitating \textit{Numeration}-like pre-selection devices (Chomsky 1995:189, §2.2.2).

We can conclude from this, that from a top-down perspective, \textit{Merge} reduces to \textit{lexical insertion}.

Other empirical advantages, proceedings from left to right using extended projections, come from agreement handling:

\begin{equation}
\begin{array}{ccc}
(28) & il & gatti & (Italian) \\
& [\text{number=sing the}] & [\text{number=plur cats}] & \\
\end{array}
\end{equation}

Assuming that the interpretable features (e.g. features with a semantic interpretable content, like $\boxdot$-features on N) are on the head (that is exactly N in the “DP” extended projection) and that agreement percolates all the way through the expended projection, this explains how, in the example (28), the determiner (and any potential adjectival form modifying the noun) has to agree with it. The processing order, on the other hand, could explain why agreement on the verb (triggered by arguments in certain configurations) is required when the direct object is cliticized before the verbal head (29).b vs. (29).b’, and not when it is realized after (29).a’ vs. (29).b (cf. Kayne 1991):

\begin{equation}
\begin{array}{ccc}
(29) & a. Gianni ha mangiato & \text{una mela} \\
& \text{John has} [\text{gender=male eaten}] & [\text{gender=fem an apple}] \\
& \text{a’.*Gianni ha mangiata} & \text{una mela} \\
& \text{John has} [\text{gender=fem eaten}] & [\text{gender=fem an apple}] \\
& b. *Gianni l’ha & \text{mangiato} & \text{(una mela)} \\
& \text{John \ it(an apple) has} [\text{gender=male eaten}] & [\text{gender=fem an apple}] \\
& \text{b’. Gianni l’ha} & \text{mangiata} & \text{(una mela)} \\
& \text{John \ it(an apple) has} [\text{gender=fem eaten}] & [\text{gender=fem an apple}] \\
\end{array}
\end{equation}
I think this is a crucial difference with respect to standard Minimalism worth to be stressed: what emerges from the discussion in the last paragraphs is that we can dispense the grammar from the “simplest” function that “takes two elements $\alpha$, $\beta$ already constructed and creates a new one consisting of the two” (Chomsky 2001:6), i.e. *Merge*, since this solution is empirically inadequate (unless constrained as suggested by Stabler, through selection, Merge would freely create structures that will be discarded at the interfaces) and computationally too expensive (creating ungrammatical constituents that have to be filter out makes the problem space, hence the complexity, growing boundlessly). What should be preserved here is the “sequential” nature of the operation: lexical insertion proceed evaluating feature by feature; this gives us the “left-to-right” orientation of the derivation, that would determine a strict right-branching phrase structures (as predicted by Kayne 1994), unless other devices be evaluated during the process of phrase structure building.

Here Lexical Insertion can be thought of as an information-combining operation among feature structures (similar to the unification algorithm used for instance in HPSG, cf. §1.2.1) and it directly subsumes both Agree (Chomsky 2001:12, §2.2.2) and Valuing (Fong 2004, §2.3.3).

Another non-trivial difference, is that Lexical Insertion, against Merge, is not recursive. We should look, then, somewhere else for the source of (necessary) simple recursion that allows us to make an infinite use of finite means.

### 3.3.3 Directionality of Movement and Relativized Minimality

Let me now focus on the analysis of what encodes Movement. Following Stabler’s formalization, *licensees* and *licensors* are symmetric classes, formed by the very same set of features, corresponding to the *interpretable/uninterpretable* distinction that Chomsky (1995) introduces to justify the fact that uninterpretable features have to be deleted before reaching the Conceptual-Intentional interface (the same idea lies behind the probe-goal approach, Chomsky 2001, §2.2.3). However, as Rizzi (2004)
suggests, the only thing we minimally need to identify a goal (the element to be moved in a probe headed position) would be the feature itself; on this line, we could get rid of the distinction –f / +f, using just f. This idea will be pursued here: the licensors hierarchy, per se, “triggers movement”, in the sense that a lexical element has to occupy the relevant functional projection in order to legitimate a related licensor feature.

First thing we should remark is that from the top-down, left-right derivational perspective, dispensed by the –f/+f (uninterpretable/interpretable) technology, features inspection in A′-movement will first evaluate the left-peripheral positions (i.e. the functional/criterial features that are, in classical terms, the landing site for movement), then all other intermediate positions (if any), and only at the end, the lowest position of the chain that is the thematically selected one.

Keeping this in mind, before discussing what could trigger such non-local dependency, let me discuss another empirical (related) necessity, that is, how to capture Relativized Minimality effects. As I briefly mentioned before (§2.3.2) Stabler’s definition of movement is not sufficient to predict the correct locality constraints since only feature identity would “block” a non-local dependency; remember that locality constraints on movement are not just a matter of identity of features, but rather of classes of features (§2.2.4): all elements belonging to a specific intervener class are potentially deadly interveners for movement or chain formation between elements belonging to the same class. In this sense, the kind of featural intervention we need to formalize here is on the lines of Starke (2001) or Friedmann et al. (2009): recapping Rizzi’s (1990) idea of structural intervention, a non-local dependency between X and Y is blocked if Z structurally intervenes and it is of the same kind of X (and Y);

(30) *what do you wonder who you call _?
    X          Z  Y

148
Assuming that in the above configuration Z should enter the non-local dependency with X, before Y, this explains superiority effects (Chomsky 1973) and subsumes other constraints like Minimal Link Condition (Chomsky 1995:181). As discussed in chapter 2 ((42), reported below), Starke explains the selectivity of the intervention by describing features as hierarchically organized, A (X and its copy Y in (30)) and B (Z in (30)) as selectively distinguishable depending on their featural makeup:

(42) a. *[a A] ... Bα ... [α A] *[wh, what] did [wh, who] buy <[wh, what]>? 
   b. [αβ A] ... Bα ... [αβ A] [wh, d-linking which book], did [wh, who] buy <[wh, d-linking which book]>? 
   c. *[a A] ... Bαβ ... [α A] *[wh, what] did [wh, d-linking which student] buy <[wh, what]>? 

Adopting Friedmann et al. (2009) terminology, let me call (42).a identity, (42).b inclusion and (42).d disjunction. Following their work, these distinctions seem to be relevant not only for explaining ungrammaticality, but also for understanding processing difficulties in establishing different kind of A’-dependency:

(31)a. The reporter_i [that the senator attacked _] disliked the editor. 
   b. The reporter_i [that you attacked _] disliked the editor. 
   c. The reporter_i [that _ attacked the senator] disliked the editor.

According to Gordon at al. (2001 a.o.) (28).a is harder than (28).b, which is harder than (28).c. 68 This is because in (28).a and (28).b, but not in (28).c, an intervener occurs between X (“the reporter”) and Y (the selected thematic position where “the reporter” must be interpreted). On the other hand, “the reporter” and “the senator”

68 The fact that Subject-headed Relative Clauses of these kinds are harder than Object-headed Relative ones has been shown systematically using self-paced reading experiments (King & Just 1991 a.o.), probe-task paradigms (Warner & Maratos 1978 a.o.), brain activity analysis (Just et al. 1996 a.o.) and eye-tracking techniques (Traxler et al. 2002).
share more features than the ones shared between “the reporter” and “you”: in the first case, there is a “lexical restriction” (as Friedemann et al. 2009 call it) that is not present in “you” (28).c. This is why (28).a fully falls under the inclusion case and (28).a is a more close to an identity case, even though some sort of relative marker distinguish the two items so as to allow the dependency to be established avoiding ungrammaticality. Notice that in every case, only relevant full features matching, where “relevant” means “triggering movement” (Belletti & Rizzi 2012), causes full ungrammaticality. A critical assumption here it is however necessary, since the lexical restriction should be “nested” within the DP shell and not evidently accessible to motivate any sort of intervention, unless some percolation of features (or inspection of selection requirements, as Belletti & Rizzi 2012 suggest) would allow. It is important to notice here, that the extended projection idea allows us to “percolate” this restriction up to the node that qualifies an element as intervener, moreover making accessible other nested features that seem to interfere with the dependency, like animacy, as discussed in the similarity-based approach (Gordon et al. 2001, 2004).

3.3.4 Move is triggered by unsatisfied local dependency

Going back to Stabler’s formalization, a potential problem in his grammar is related to the deletion of the licensee features when the element bringing them is subject to successive cyclic movement:

(32) John seems \( t_{john} \) to be \( t_{john} \) happy

In (32) the selected features on John (\([D N]\)) have to survive after merge. The proposed solution, namely that these features can optionally be deleted is neither satisfactory (it would cause non-determinism) nor explanatory (no generalization could be captured). Another related inefficiency is directly associated to the mechanism driving movement that deletes features during the derivation by pairing
-f/+f features: as we speculated before (§3.3.3), there are reasons to believe that this is not the most minimal hypothesis.

Here I suggest that both problems can be solved by inverting the directionality of the Move operation; as informally anticipated in §3.3.3, processing a sentence forces a left to right feature analysis that first brings to the hearer/speaker’s attention the element in its highest structural position. In order to exactly determine this position the hearer should recover the potential licensor(s) possessed by the item, either by inspecting morphological markers (that is the word shape, Bresnan 2001, e.g. case features and maybe prosodic cues) or/and inferring it from the adjacent elements (minimally by inspecting the next item). From a generation perspective, the speaker, again, should first decide to lexicalize the head of the chain (higher position) in a functional position, then the tail of it (selected position).

In both cases, I assume that non-local dependencies are triggered by the fact that features that are lexicalized in the highest position are not expected there. This happens, for instance, when a wh- feature, expected to be in the left-peripheral position in languages like English, is lexicalized by items like “who” or “what” that brings from the lexicon other features that qualifies these items as argument (e.g. [+D N] that is a determined nominal constituent). These features keep the item that is inserted on the head of a potential chain “on-hold”, that is, the derivation should remember the set of unexpected features up to the point of the computation in which the item can be re-merged, since properly selected/expected.

More formally, we can define move as follows:

(33) Move (definition)

i. an item $[f \alpha]$ (either a lexical item or a phrase) introduced in the derivation to lexicalize a feature (bundle) $f$, if bearer of unexpected feature (cluster) $g$ that must be selected, must be stored in memory as $[g \alpha]$. 

151
ii. as soon as a head \([\alpha \beta]\) is properly introduced in the derivation, if \(\beta\) selection requirements match the feature of \(\alpha\) stored in memory, \([\beta \alpha]\) must be re-merged in the selected position before any other lexical insertion.

This definition of movement formalizes the idea that a “moved” element “stands in memory” as far as it is used to fill (interpret) the base position where it is selected (this implements the active pattern idea discussed in §1.2.2). Assuming that the memory structure is a sort of stack (Last In First Out memory) we can capture nested dependencies as shown in the examples below (35).
For sake of simplicity, I will discuss here neither head movement nor derivational morphology issues. Assume “called” is a lexical item, endowed with these features (+T, V, =DP, =DP). (+T, V) both trigger the insertion in the +T functional position and also check the (V) lexical (head) position. See next note for +S.
(35) Who did John call?\textsuperscript{70}

\textsuperscript{70} +S is a criterial position related to the sentential subject: in SV languages, this forces the first insertion of the subject in the structure. On a lexical item, this feature should be interpreted as a (null, in English) case marker.
In principle we do not need anything else to account for Relativized Minimality effects: ungrammaticality in cases of feature identity, (42).a is explained by the fact that the element Z is more prominent in memory than X then it enters first the dependency in which Y should be involved; on the other hand, in case of inclusion, and/or disjunction assuming that the memory, as the lexicon, is searchable by feature, we can expect the relevant item to be identified much more easily in disjunction cases than in case of inclusion. This prediction is empirically testable and it is indeed coherent with the similarity-based account of memory interference (Gordon et al. 2001).

Before concluding, we must assume that whenever a dependency is unexpected, and some features are stored in the memory buffer, this should lead to ungrammaticality unless, at some point, a proper selector is introduced in the derivation and the unsatisfied dependency “discharged”. To guarantee this, we need to state a success condition:

\[(36) \text{Success Condition} \quad \text{(definition)}\]

\[\text{a sentence is grammatical if only if at the end of the derivation, memory is emptied of any unsatisfied dependency.}\]

This condition has important consequences on the understanding of crucial phenomena, like Strong Island Conditions (Ross 1967) as I will discuss in §4.

3.4 Complexity Issues

As I mentioned in §1.4, the complexity of a problem is an expression of the resources (essentially time and memory) needed to solve the problem. More precisely, it is a function of the size of the problem, determined at least by three factors:

\[(37) \text{Complexity factors:}\]

i. the length of the input \((n)\);
ii. the space of the problem (all states the system can attain by correctly applying any legal rule);

iii. the algorithm used to explore this space.

while (37).i is largely independent from the grammar, (37).ii and (37).iii are strictly determined by it. From a Minimalist perspective, (37).ii is mostly determined by the lexicon and by the structure building operations Merge and Move; (37).iii is usually assumed to be a bottom-to-top algorithm (namely an algorithm that starts building structures from the inner most verbal shell, adding piecemeal more peripheral elements) plus some economy conditions (Shortest Move/Attract Closer, Earliness Vs. Procrastinate, Merge preempts Move etc.).

The problems we would evaluate within this framework in terms of complexity are the parsing and the generation ones (§3.1.2). Both problems have two distinct components to be explored: a part of lexical ambiguity resolution (“find the relevant set of lexical items Lex” compatible with $\pi/\sigma$) and a part of structural mapping (from precedence to dominance and vice versa). These problems are difficult to solve because of a theoretically non-univocal mapping between phonological features and words (homo-phony/graphy, polysemy, synonymy etc.) and because of discontinuous constituents (non-local dependencies) that cause structural ambiguity. The goal of using phases (Chomsky 1999) is then on the one hand to reduce the space of the problem, on the other to make the complexity function independent from the input length.

3.4.1 The Complexity of Ambiguity
Considering both lexical and semantic ambiguity in parsing (§3.2.1), the first (sub)problem can be stated as follows:

(38) Ambiguity problem in parsing (definition)

given an input $i$ of length $n$, composed by $\pi$ features grouped by words, and

156
a number $c$ of possible Part-of-Speech (PoS), assign to each word from the input at best one PoS, if possible. If not reject the input.

The complexity of the problem, considering a brute force algorithm to solve it, is at worst $O(c^n)$ (assuming that any word could be ambiguous among all PoS).

A more realistic complexity order can be guessed by inspecting dictionaries or corpora. For instance, using Wordnet (Miller 1995), the ambiguity factor 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 word occurrences seem to be ambiguous and most of them are ambiguous between two PoS; then we could underestimate the ambiguity factor up to 40%, producing an order of complexity of $O(1.4^n)$.

However, this is clearly not enough yet for the problem to be tractable: analyzing a text would imply processing hundreds of words; the exponential combinatory of the problem still 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 such a context

---

71 Categories (or PoS) have to be intended as “indices” pointing to fully specified items in the lexicon. This way the very same problem can be stated from a generation perspective simply changing “π features” in “σ features”.

72 $\text{ambiguity factor} = \frac{\text{synsets}}{\text{lexical entries}}$

73 With just 50 words to be disambiguated we could evaluate up to about 20 millions of possibilities.
cannot be arbitrarily fixed (e.g. *n*-grams approach, Jurafsky & Martin 2000): the length of a grammatical phrase containing ambiguities can be arbitrarily long (in principle it can be infinite), then fixing it once and for all would not be heuristic. It is also implausible to reduce the “structurally defined” context to simple local selection as shown below:

(39)a. The [dogs] run ([D the] selects [N dogs])
   b. Mary [dogs] me ([DP Mary] does not select any [V dogs])
   c. Adverbials cannot select all possible lower adverbials and/or the verb (this would raise the problems discussed in §3.3.1).

A more adequate solution is to define the *phase* as the “largest context” within which an ambiguity can be solved. As introduced in §3.3.1, I assume that each *phase* originates in a lexical head \( (N \text{ or } V) \), which then projects and dominates the functional licensors features, as shown in (24). Note that the phase turns out to be dimensionally bounded in length (maximum number of precedence relations) and in depth (maximum number of dominance relations) because of the following constraints:

- given a fixed hierarchy of licensors features (cf. §2.2.4), a phase contains at worst \( k \) functional elements;
- a phase contains exactly one projecting lexical head (by definition of phase, (25));
- by assumption (Pesetsky 1982), each lexical head selects at most three phases (subject, object and indirect object);

At this point, the complexity order within each phase would be \( O(1,4^{k+4}) \) (where 1,4 is the most optimistic ambiguity factor based on dictionaries and on corpora inspection): the crucial point 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 complement represent a new phase opened within another phase) would produce an increase of the complexity order of $O(1.4^{p(k+4)})$ 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 by many processing experiments when the subjects have to parse/generate sentences with structures that clearly show a difficulty in “keeping unsolved ambiguities” (e.g. “Buffalo buffalo buffalo Buffalo buffalo” is a well-formed sentence in English but it is extremely difficult to parse unless we already expect its structure: “bison from Buffalo (NY) confuse (other) bison from Buffalo”). Thus, the more phases we open (at the same time) the more difficult the problem will be.

In sum, the first desideratum met by the introduction of the notion of phase is to restrict productively the context where certain elaborations, such as ambiguity resolution, could take place.

3.4.2 The Complexity of Non-Local Dependencies

From a parsing perspective, considering 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): among $n$ items, in fact, we should define $n-1$ (immediate) dominance 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 ($A\times B$ or $B\times A$).
Even if we limit ourselves to the discussion of simple movements\(^{74}\) the complexity increases significantly: at worse, any item could have been potentially moved from/to any lower (licensed/selected) position, so that 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\} and so on. Then (((n-1)+1)·(n-1))/2. This is clearly not a satisfactory result, since the growing rate of any of these functions would make the evaluation problem quickly intractable\(^{75}\). Notice, however, that there are many empirical constraints on movement (captured by the standard theory) that could significantly affect the space of the problem:

(40)  
\[
\begin{array}{c}
A/B \\
B & A \\
A & (B)
\end{array}
\]

\(\ast\)re-merge
\(\ast\,[\text{VP Mary kisses } t_{\text{Mary}}]\)

\(^{74}\) Also without considering successive cyclic movements the argument is sound and significant.

\(^{75}\) We do not need to argue that complexity is, in fact, even worse than that, since unpronounced elements can enter dominance relations (possibly without being linearized) and there is no way to determine how many empty elements could be present, in principle, in a sentence of length \(n\).
b. *self-merge
   *kisses [VP Mary \( t_{kiss} \)]

\[
\begin{array}{c}
  \text{B} \\
  \text{B} \\
  \text{B} \\
  \text{B}
\end{array}
\]

\[
\begin{array}{c}
  \text{A} \\
  \text{(B)}
\end{array}
\]

c. *move and project
   no landing site would be available before
   the movement if the moved element would
   project; moreover this option would
   violate the assumption that movement is
   triggered by uninterpretable features on a
   probe (Chomsky 1999); in this case probe
   and the goal would trivially coincide
   ("target project condition" Chomsky

\[
\begin{array}{c}
  \text{B} \\
  \text{B} \\
  \text{B} \\
  \text{B}
\end{array}
\]

\[
\begin{array}{c}
  \text{C} \\
  \text{A} \\
  \text{A} \\
  \text{B}
\end{array}
\]

d. *lowering
   following Kayne (1994) any apparent
   instance of lowering (Right Node Raising,
   Extraction Across the Board etc.) could be
   accounted for in terms of remnant
   movement ((126.e) is however implied by
   (126.a))
e. *movement from nothing
the base position on a chain should be selected. An empty element cannot select anything

f. 'orphans formation, head movement
verb movement

g. 'movement from functional positions
adverbial focalization, successive cyclic movement
Even though these restrictions are defined in a bottom-to-top perspective, the very same constraints can be obtained also in a top-down framework. These restrictions unfortunately do not significantly reduce the growing rate of the function, since the only formal restriction that they predict is that an element cannot be moved from a position to the right of the next element (namely at least two elements should be interposed between the landing site position and the originating one).

The remaining intractability in parsing can however be improved upon by adopting the idea of phase: we have assumed before (§3.3.4) that movement can be detected by the presence of an element that is not selected; in this case, this element would stand in “memory” as far as another element selects it. Following Chomsky (1999), let us assume that movement takes place within the phase: then, given a limited context corresponding to the phase boundaries, either we find the selecting element, so we can reconstruct the moved object in its base position, or else if we do not find any selecting element, the expectation to find a selecting sister for the unselected element will be projected on the lower phase (leaving an intermediate trace on the “left-periphery” or this lower phase). And so on.

This derivation can be abstractly represented by the chain idea (Rizzi 2004):

\[
(41) \text{criterial} \quad \text{(intermediate traces)} \quad \text{selection}
\]

A criterial position is the position where the element is interpreted as (roughly) a scope-bearing element, via some interpretable licensor feature. Crucially an element cannot be moved from a criterial position: meeting a criterion freezes the element in place (Rizzi 2004:11) as shown below:

(42)a. *Who, do you wonder \([_{CP \ t_i} \ [_{TP \ t_i} \ did \ [_{VP \ t_i} \ call \ you ]]])?  

b. *Who, did \(t_i\) call you do you wonder \(t_i\)?
From our perspective, a criterial position is the position from where an element is inserted into the memory buffer. Selected positions, on the other hand, are those positions where an element can be discharged from the memory buffer; this happens because of a lexical head that selects the features of the element that has been stored in the memory buffer. Between criterial and selection position, there could be intermediate positions where the element is “copied” before reaching its (selected) final destination. The properties of these intermediate traces are:

- 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 uninterpretable features, §2.2.2).

These properties strengthen the idea that intermediate traces in the “left periphery” (edge, Chomsky 1999) of the phase serve as “memory refresh” (phase balance, Felser 2001) to make long distance dependencies possible. This empirical generalization produces a remarkable reduction of the complexity problem: in fact, for any moved element, we would have at most two possible landing site positions within a phase (a left-edge position, used as memory refresh, and one selected position). Then, given a limited context, either we introduce an extra dominance relation and we license the moved element within the current phase\(^76\), or we put it aside and consider this element within the next phase to be processed. This procedure can be reiterated recursively (and it is potentially non-directional). Only when an extra dominance relation is introduced within the currently processed phase, the element is frozen in place and cannot move further. Assuming \(k\) to be the

\(^76\) For the present discussion the actual position/status of this dominance relation is irrelevant.
fixed maximal size of the phase, then for any phase the number of dominance
relations required would be, at worst, $2^{2k}$ (in case anything would have been
moved\(^7\)). The complexity order of the problem, considering any dominance as
ambiguous, would be $O(2^{2k})$.

Interestingly, if we assume that we can process phases “in parallel”, we obtain an
asymmetry in complexity growth: processing a phase before having completed the
previous one leads to a duplication of the number of possible dominance relations
and so on as long as new phases are opened. The 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 non-overlapping partition of the input; in particular, the
number of lexical items in $n$ would determine the number of phases (which could be
$n$, at worst).

\(^7\) Phases are non-overlapping partitions of the input since every node univocally belongs to a
single phase and a lexical item is fully inserted/lexicalized in a single (subset of strictly
adjacent) node(s).
<table>
<thead>
<tr>
<th>Functions:</th>
<th>( N^o ) of relations to be evaluated</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( n=6 )</td>
</tr>
<tr>
<td></td>
<td>((p=2, k=3))</td>
</tr>
<tr>
<td><strong>Any option</strong></td>
<td>( \frac{n(n-1)}{2} )</td>
</tr>
<tr>
<td><strong>Nested Phases</strong></td>
<td>( 2^{p^2k} )</td>
</tr>
<tr>
<td><strong>Sequential Phases</strong></td>
<td>( p \cdot 2^{2k} )</td>
</tr>
</tbody>
</table>

Note that long distance dependencies within a phase would not produce any remarkable effect on the complexity of the problem; discontinuous constituency relations among phases would increase the complexity of the problem in a linear way if each phase is completed before the next one starts (sequential phases), whereas there is an exponential increase of complexity whenever we open a phase before having closed the previous one (nested phases). We will see in §4 that this expected increase of the complexity, caused by the movement of elements across open phases provide us with an account of the notion of (strong) islands (Huang 1982).

3.4.3 How Phases constraint Non-Local Dependencies

Summarizing, a phase, from a Top-Down perspective is an extended projection (Grimshaw 1991) of a lexical head continued by complement projections organized

\[^{\text{78}}\]p is the number of phases in the input, \( k \) is the maximum size of the set of items per phase. Both numbers can be arbitrarily fixed to any finite natural number.
as larsonian shells (Larson 1988). What we obtain is the following schematic structure for a phase projected by a lexical V head:

(43)

The verbal head projects upward, dominating (in a label-free grammar, Collins 2001) every functional projection that precedes it, and downward, dominating its complements. This seems to be a step back with respect to Abney’s influential work (Abney 1987), but in fact, assuming standard selection among functional projections would obscure the intuition that they are all part of the same phase-

---

79 Larsonian shells are projected in accordance with selectional requirements that should be encoded on the verbal head in the lexicon. Mutatis mutandis, we can assume that nominals project a parallel structure. For an attempt to draw some parallelism between nominal and verbal functional domains see Ogawa 2001, Grohmann 2003 and Laenzlinger 2005 among others.
projection and they are related to one and the same lexical head, solving moreover the empirical problems discussed in §3.3.

As discussed in the previous paragraph, a maximum finite number of elements can be guaranteed per phase: let us say at most \( k \) functional elements plus at most three complements\(^{80}\). It is clear, however, that every element of such a phase can be constituted by another phase: for instance, a DP in a preverbal argumental position would be a nominal phase embedded within a verbal phase; the same holds for PP adjuncts which should be attached/related to a precise functional specification: they are nominal phases within verbal phases\(^{81}\) (and so on, in a potentially recursive fashion). From a left-to-right perspective, then a phase corresponds to a processing domain that can be “interrupted” before completion by the “expansion” of another phase. From this reasons I will call the “interrupting” phases *nested phases* and the last selected phases the *sequential phase* (Bianchi and Chesi 2006):

---

\(^{80}\) If we accept Borgonovo and Neeleman’s idea (2000), namely that the predicated event can take arguments too, the number of arguments in a phase can be bigger, yet crucially finite.

\(^{81}\) From this perspective, the preposition is a case marker that take a case peripheral position above the classic DP (e.g. \( K \) position discussed in Ogawa 2001).
Since expanding nested phases would quickly lead to an exponential growth of the number of possibilities to evaluate (as discussed in §3.4.2 in case of Non-Local Dependencies) I assume that the grammar excludes certain options, like evaluating all kinds of dependency within nested domains.

In order to implement this “complexity avoidance” strategy, I assume that the content of the memory buffer is phase-local and, at best, can be used only through sequential phases. This amounts to say that a non-local dependency must be satisfied within the current phase or in a properly selected phase, but not in a nested phase:

(45) Constraint on the usage of Memory Buffers (M-buffers)

i. M-buffers are phase local: an item is moved (since bearer of unexpected features) in the M-buffer of the phase in which has been introduced, and must be discharged in a selected position of this phase;

ii. Nested phases can neither discharge nor populate the M-buffer of a superordinate phase;

iii. If at the end of the phase processing there are still elements pending in the M-buffer, its content must be discharged in the M-buffer of the last
selected phase (the sequential phase) if any (items pending in the M-Buffers at the end of the derivation lead to ungrammaticality; Success Condition (36)).

We can now understand that phase expansion is indeed the only recursive component of our grammar, and it is strongly constrained in terms of non-local dependencies by (45): selection features on the lexical items create the expectations for a certain phrase structure to follow. This creates the main recursive branch of the tree, that, because of the notion of sequential phase is, cross-linguistically, to the right, i.e. after the selecting head\(^{82}\).

\(^{82}\) There are two issues related to this point which I do not have space to discuss: the first is the apparent counterexample provided by head-final languages like Japanese (see Choi & Yoon 2006). The second is that selection has the ability to introduce treelets, as in TAG; this has been regarded as a problem, correctly to my opinion, in Boeckx & Piattelli-Palmarini’s (2007) work: they argue that having treelets instead constituents resulted from binary Merge is not explanatory (how treelets are built and why they have this shape?). The idea we should pursue here is that local selection should be semantically driven: constraints about how comes that some merges are successful while some others are not would bring to the very same axiomatic conclusion: e.g. \(T\) is selected by \(C\), by definition. More investigation on this point is necessary.
So far, I pointed out that the model presented in this book is both cognitively plausible (chapter 1) and computationally efficient (chapter 3). These results are greatly related to some decisive assumptions made by recent formal and computational linguistic theories (chapter 2) that have been incorporated, as transparently as possible, within the model sketched in these pages: for instance the notion of *derivation by phase* (Minimalist Program, §2.2.2) and the idea of a *functional universal hierarchy* (Cartographic Approach, §2.2.4) besides a precise formalization of *minimalist grammars* (Stubler 1997, Fong 2004).

Some important modification of these ideas however has been shown to be necessary in order to meet “interface conditions”, or better, to be used by performance algorithms (or tasks) such as *parsing* and *generation* (§3.1.2). The directionality of the derivation from left-to-right (as firstly proposed in Phillips 1996), top-down seems to be the only viable option we have to meet a “token transparency” (Chomsky & Miller 1963) requirement. This modification forced us to review critically the main structure building operations such as Merge, Move and the Phase idea. Merge reduced to *lexical insertion*, while both the concept of Move (with the use of a memory buffer helping building non-local relations from left-to-right) and the notion of Phase (conceived as a top-down expectation, driven by lexical selection) has been significantly modified. At this point, these modifications should be enquired also in terms of descriptive adequacy (*universality* requirement, in the Introduction). In this chapter then I will summarize the precise specification of the grammar apt to be used both in parsing and in generation (§4.1). Then I will try to present some arguments showing that a wide empirical coverage has been attained using this model (§4.3).
4.1 Parsing and Generating sentences using the same Grammar

As Stabler’s Minimalist Grammar, also the top-down grammar introduced in these pages, that I will call Phase-based Minimalist Grammar (PMG), can be described as a 4-tuple \{V, Cat, Lex, F\} such that:

- \(V\) is a set of non-syntactic features, \((\pi \cup \sigma)\) where \(\pi\) are phonetic features and \(\sigma\) semantic features;
- \(Cat\) is a set of syntactic features, such as \(Cat = (\text{base} \cup \text{select} \cup \text{licensors})\) where \(\text{base}\) is a set of two lexical categories \(\{V, N\}\), \(\text{select}\) specifies the selectional requirement of the head in terms of \(\{\text{base} \cup \text{licensors}\}\);
- \(\text{licensors}^{83}\) is a pair of ordered sets of elements expressing the functional features associated any lexical head \(\{V = < +\text{Force}, +\text{Top}, +\text{Foc}, ... +\text{C}, +\text{Mood}_{\text{speech}}_{\text{act}}, +\text{Mood}_{\text{evaluative}}, ... \text{T}_{\text{past}}, ... \text{Asp}_{\text{completive}}>\), \(N = <\text{K}, ... \text{D}, \text{ordinal, cardinal, ... size, length, height ... color, nationality, material}>\)
- \(Lex\) is a finite set of expressions built from \(V\) and \(Cat\) (the lexicon); it is divided in two subsets: \textit{lexicalized} items, i.e. items with \(\pi\) or/and \(\sigma\) content, and \textit{unlexicalized} ones (phases), i.e. items with only \(Cat\) content.
- \(F\) is a set of three partial functions from tuples of expressions to expressions \(\{\text{merge, move, phase}\}\) where:

---

83 Here I will not discuss the hierarchical structure of the licensor features, i.e. their subclasses. They are however necessary to account correctly for Relativized Minimality effects (Rizzi 1990) (e.g. \(Q = \{\text{Wh}, \text{Neg}, \text{measure, focus, ... }\}, \text{Mod}(_\text{ifier}) = \{\text{evaluative, epistemic, Neg ... celerative, measure ... }\}... \text{see}\ §2.2.4).
**merge** given an unlexicalized feature \( f \), lexicalize it by substituting it with a lexical item bearing \( f \) and mark it as lexicalized. First use the lexical item, if any, in the M-buffer of the current phase, deleting it from the M-buffer, then inspect the lexicon. Merge is greedy, so if a sequence \( f \) and \( g \) can be lexicalized with the same lexical item, lexicalize both;

**move** if a feature \( m \) is introduced in the computation, as a result of the lexicalization of a feature \( f \), but does not lexicalize anything, put it in the most prominent position of the M-buffer of the current phase;

**phase** given a select feature, project the minimal set of dominance relations that satisfy the selection; any selected phase has its own M-buffer; only the last selected phase inherit the content of the M-buffer of the projecting phase.

Let us see a toy grammar in action by deriving the example (46).a with the lexicon (46).b divided in *unlexicalized phases* and *lexicalized items*:

(46).a. What did John buy?\(^{84}\)

---

\(^{84}\) Thanks to Valentina Bianchi for the Box-Notation below that allows us to describe the derivation in a compact way: \( P_n \) are phases, \( M_n \) M-buffers, box inclusion indicates nesting, adjacent boxes indicate sequential phases. Gray arrows indicate movement steps.
The steps of the derivations expressed in the diagram (46).a are described below:

1. a default verbal phase is projected (P₁):
   \( \text{phase(Wh- question) = } [+\text{wh} +\text{T } +\text{S } +\text{V}] \)

2. since the verbal phase is interrogative, this functional feature has to be explicitly marked; in English this can be done by merging the relevant wh-element within the specific “criterial” position. That is how \([+\text{wh} +\text{D N } \text{what}], \text{phase } P₂ \text{ (computed as a nested phase), is introduced in the derivation:} \]
   \( \text{merge}( [+\text{wh} +\text{T } +\text{S } +\text{V}], [+\text{wh} +\text{D N } \text{what}]) = [+\text{wh} [+\text{wh} +\text{D N } \text{what}] +\text{T } +\text{S } +\text{V}] \)

3. since \(+\text{D N }\) features are unexpected/unselected \([+\text{D N } \text{what}]\) is inserted (step 1) in the memory buffer (M₁) of the matrix V-phase (P₁):
   \( \text{move}( [+\text{D N } \text{what}]) = \text{M₁:<[+\text{D N } (\text{what})]>} \)

4. \textit{did} is compatible with a tense functional specification of the matrix V-phase, then licensed in this position:
   \( \text{merge}( [+\text{wh} [+\text{wh} +\text{D N } \text{what}] +\text{T } +\text{S } +\text{V}], [+\text{T } +\text{did}]) = \)
   \( [+\text{wh} [+\text{wh} +\text{D N } \text{what}] +\text{T } [+\text{T } +\text{did}] +\text{S } +\text{V}] \)

5. since \([+\text{S } \emptyset]\) has a vacuous semantic content, the lexicalization of \(+\text{S }\) requires two steps:
   \( \text{merge}( [+\text{wh} [+\text{wh} +\text{D N } \text{what}] +\text{T } +\text{S } +\text{V}], [+\text{S } [+\text{S } \emptyset ] +\text{D N }]) = \)
   \( [+\text{wh} [+\text{wh} +\text{D N } \text{what}] +\text{T } [+\text{T } +\text{did}] [+\text{S } [+\text{S } \emptyset ] +\text{D N } +\text{V}]) \)
   \( \text{merge}( [+\text{wh} [+\text{wh} +\text{D N } \text{what}] +\text{T } +\text{S } +\text{V}], [+\text{S } [+\text{S } \emptyset ] +\text{D N }]) = \)
   \( ([+\text{wh} [+\text{wh} +\text{D N } \text{what}] +\text{T } +\text{T } +\text{did}] [+\text{S } [+\text{S } \emptyset ] +\text{D N } +\text{V}]) \)

\[85\] It is fair to assume that aux-subject inversion is decided (as parameterized option) at this level; the select feature “wh-question” is encoded in the grammar.
6. \([+D N \text{ John}]\) (phase \(P_3\), again computed as a nested phase) is introduced to satisfy a subject-criterial (in the sense of Rizzi 2004) requirement (functional specification of \(P_1\)) and moved in the memory buffer since \([+D N]\) are unselected (step 2)

\[
\text{move}([+D N \text{ John}]) = M_1: [+[D N (\text{John})], [+D N (\text{what})]]
\]

7. then \([+D N] \rightarrow [+D N] \text{ v buy}\) is processed as the head of the matrix V-phase \((P_1)\). Since it has two selection requirements to be satisfied (an agent and a patient, both N-phases with features \([+D N]\)), these select features will project two phases, \(P_4\) e \(P_5\).

\[
\text{merge}([+\text{wh} [+\text{wh} +D N \text{ what}] +T [+T \text{ did}] [+S [+S \emptyset] [+D N \text{ John}]]]) \text{ V },
\]

\[
[\text{V} [+D N] [+D N \text{ buy}]) = [+\text{wh} [+\text{wh} +D N \text{ what}] +T [+T \text{ did}] [+S [+S \emptyset]]
\]

\[
[+D N \text{ John}] \text{ V} [+D N] [+D N \text{ buy}])
\]

\[
\text{phase} (= [+D N]) \text{ (per two)} = \ldots [\text{V} [+D N] [+D N]]
\]

8. \(P_4\) is a nested phase and will be lexicalized by (re-)merging \(P_3\) (the first accessible element in \(M_1\)); this element will be removed from \(M_1\)

\[
\text{merge}(\ldots \text{V} [+D N] [+D N])], M_1([-D N (\text{John})) = \ldots \text{V} [+D N] [+D N (\text{John})])]
\]

9. \(P_5\) is the last Phase: since it is the last selected argument of the matrix phase, this will be the sequential phase and will be lexicalized by merging the last pending element in the memory buffer, i.e. \(P_2\):

\[
\text{merge}([-D N], M_1([-D N (\text{what}))) = [-D N \text{ what}]
\]

I would like to stress that, despite its apparent similarity with a parsing algorithm, this is not a mere parsing strategy but it is the formal implementation of the derivation. From a \textit{chain}-based perspective, the solution discussed for dealing with non-local dependencies (i.e. movement) simply shifts the focus from the tail-position (selected position, always lexically derivable), to the head-position of a chain (scope position, dynamically determined by the speaker in a multiple clausal
context). This shift is necessary in order to make the movement operation deterministic without any look-ahead device for intermediate steps as shown by the following schematic derivation:

(47) Who do you believe (that Mary said that ...) that everybody admires?

Movement in (47) is driven by the fact that unexpected thematic features ([D/Q,N]) are introduced in the derivation to satisfy a criterial requirement. This forces this element to stand in memory until a lexical head properly selects them as argument. The inheritance mechanism through sequential phases guarantees that any arbitrarily distant dependency, laying on the right recursive branch of the tree (and properly selected), can be established without need of any uninterpretable feature, hence solving any indeterminacy and/or violating the finitary nature of the lexicon by postulating that any step of unbounded cyclic movement must be triggered by a distinct feature (§3.3).

The grammar here described is readily used in generation simply ordering the Structure building operations as follows:

1. **Phase** (project)
2. **Move** (if any element is in the memory buffer and it is compatible with the actual selection)
3. **Merge** (pick up a compatible lexical item from the lexicon, according to the selected feature(s) to be unified).
Notice that the order of Merge and Move operations is exactly the opposite with respect to the one predicted within a bottom-to-top derivation. Here in fact, Move preempts Merge. This is coherent with the psycholinguistic intuition that an element must be licensed as soon as possible (Frazier & Fodor 1978, De Vincenzi 1990 Gibson 1998) and it is also empirically sound (Richards 1999).

Despite the fact that many points need to be specified (e.g. PP attachment and, in general, syntactic ambiguity resolution), the resemblance of this algorithmic derivation with respect to the parsing algorithm proposed by Fong (2004) (§2.3.3) strongly suggests that the token transparency is much closer than for standard Minimalist grammars.

4.2 Empirical coverage, the case of Strong Island

The idea of implementing a computational model inspired to recent minimalist/cartographic research was justified by the necessity to meet universality (that is a cross-linguistic version of descriptive adequacy as explained in the introduction) and explanatory adequacy. Despite the radical revisitation of the Structure Building Operations, the very same set of relevant empirical data is captured by this model. Indeed many new empirical contrasts have been explained in the last six years, since this model began to be tested. These last paragraphs quickly show that this idea is plausible in many important ways. Then some relevant linguistic phenomena will be discussed in order to verify whether or not this model is descriptively adequate and to refine, in case, some highlighted problematic aspects.

Here I want to briefly discuss the case of strong island constraints: in SVO languages, sentential subjects of (di)transitive verbs behave like (strong, in the sense of Cinque 1990) islands (48); on the other hand, subjects of unaccusatives and passives (49) seem to allow for smoother extractions (Chomsky 2008):

(48)a. *Who did [ [close friends of I] become famous]?
   b. *I wonder what [ [reading I] would be boring]
a. [Of which car] was [the driver t] awarded a prize?
b. It was the CAR (and not the TRUCK) of which [DP the driver t] was found.

On a par with the first class of subjects, “true” adjuncts too, (50) Vs. (51), are considered to be strong islands for extraction:

(50)a. *Which concert did you sleep [during t]
b. *How did you leave [before fixing the car t]

(51)a. What did John arrive [whistling t]? (Borgonovo & Neeleman 2000:200)
b. What did John drive Mary crazy [trying to t]?

Looking at cross-linguistic variation, adjuncts (52).a but not subjects (52).c\(^6\) behave as islands in head-final languages such as Japanese (Saito & Fukui 1998):

   what-ACC John-NOM Mary-NOM bought since angry Q
   ‘What?, John is angry [because Mary bought t].’
   what-ACC John-NOM Mary-NOM bought fact-ACC problem-into making Q
   ‘What?, John is making an issue out of [the fact that Mary bought t].’
   what-ACC John-NOM Mary-NOM bought fact-NOM problem-is that think Q
   ‘What?, John thinks that [the fact that Mary bought ri ] is a problem.’

The classical solution to account for the islandhood of both subjects and adjuncts is Huang’s (1982) CED:

---

\(^6\) The sentence is degraded since it involve an extraction from a complex NP, but crucially does not contrast with (52).b (Saito & Fukui 1998).
(53) *Condition on Extraction Domain (CED)*

Extraction is only possible out of phrases which are *properly governed*, where a node $A$ is taken to properly govern a node $B$ iff

i. $A$ c-commands $B$ and no major category boundary intervenes between $A$ and $B$,

ii. $B$ is contained within a maximal projection of $A$, and

iii. $B$ is assigned its thematic role by $A$

This condition essentially predicts extractions from objects but it leaves the data in (52) unaccounted for.

Within the more recent minimalist framework, Uriagereka (1999) suggests that cyclic spell-out could explain islandhood if we take into account economy conditions on linearization: in fact, it could be simpler for the linearization system (e.g. some implementation of Kayne’s LCA) to apply to smaller chunks rather than to the whole set of terminal nodes in the structure. Subjects and adjuncts can be shipped to spell-out as independent workspaces within which linearization can target only the relevant subset of elements. The matrix sentence and its object(s?) are instead linearized/spelt-out within the same workspace. Assuming that elements included in distinct derivational workspaces cannot freely enter the linearization of other workspaces, this proposal elegantly accounts for islandhood effects.

Both solutions require non trivial modifications so as to accommodate the contrast (48) Vs. (49) and (50) Vs. (51). Also the cross-linguistic variation pointed out in (52) seems to be unexplained. More precisely, we could force the application of CED at a specific derivational point in order to allow for extraction from passives/unaccusatives (then adapting this condition to a derivational framework), so as to accommodate (48) Vs. (49); moreover, we could explain the transparent status

---

87 Remember that a similar intuition justified the formalization of the notion of phase in §3.
of some “lower” adjuncts (such as without- or after-clauses, (51)) by assuming that these adjuncts have to be somehow thematically related to the predicated event structure (on the line of Borgonovo and Neeleman 2000; see also Truswell 2007). The Japanese data, however, remain unaccounted for. The same is true of a multiple spell-out approach (where, in fact, it would be even more problematic to explain why the subjects of unaccusatives/passives should be linearized within the matrix clause and not be independently shipped out to distinct derivational workspaces as other surface preverbal subjects). The top-down perspective on strong islands is radically different from the standard view: as shown in (54), the ungrammaticality of this sentence, repeated below, does not result from the impossibility of extracting an argument from the island, but rather from the impossibility of integrating the unselected element within the phase in which it has been introduced or within another phase, that is sequential to the originating one:

(54) *Who did [close friends of _ ] become famous?

While the islandhood of subjects of (di)transitive verb follows directly, the transparency of passives/unaccusatives subjects needs a very strict application of the memory buffer inheritance mechanism ((45), §3.4.3); the definition is repeated below:
(45) **Constraint on the usage of Memory Buffers (M-buffers)**

i. M-buffers are phase local: an item is moved (since bearer of unexpected features) in the M-buffer of the phase in which has been introduced, and must be discharged in a selected position of this phase;

ii. Nested phases can neither discharge nor populate the M-buffer of a superordinate phase;

iii. If at the end of the phase processing there are still elements pending in the M-buffer, it’s content must be discharged in the M-buffer of the last selected phase (the sequential phase) if any (items pending in the M-Buffers at the end of the derivation lead to ungrammaticality; *Success Condition* (36), §3.4.2).

We can successfully derive the extraction in (49) by first discharging the preverbal subject in the thematic position projected by the unaccusative/passive phase-head; then, since this is the last selected argument, we should discharge within this projection the *wh*- element previously inserted in the memory buffer.

The contrast in (50) Vs. (51) requires us to first discuss the problem of right hand adjuncts. First observe that the Linearization Principle discussed in §3.2.2 does not allow an adjunct to be linearized to the right of the lexical head which it is a functional specification of. Notice now that the adjuncts, being non-selected (since attached to a functional specification) have to be considered genuine nested phases. Consider for instance the derivation of (55):

---

88This is a standard prediction in any theory that assume some version of Kayne’s LCA.
Ignoring for simplicity the phases $P_3$ and $P_n$, the relevant part of the derivation is that the topicalized phrase $P_2$ is stored in the memory buffer since it is unselected (step 1), but it can be discharged neither in $P_5$ (step 2) which is nested/unselected, nor in a sequential phase, since * went does not select any other phase. Thus the islandhood of adjuncts too is a consequence of computational nesting.

This solution seems to be correct, as parasitic gaps constructions show:

(55)’ [Those boring old reports], Kim filed [without reading $e_i$].

In (55)’, and not in (55), an extra gap on the matrix sentence is entitled to discharge the content of the M-buffer. The nested constituent, since unselected, can take advantage of the item stored in the superordinate memory buffer, using it “parasitically” (Bianchi and Chesi 2006, Phillips 2012), that is, without removing it from the M-buffer of the matrix clause. The intermediate status of certain adjuncts, sometimes qualifies as argument (51) some other time as truly unselected adjuncts (50) explains the contrast (Borgonovo & Neeleman 2000).

In order to elucidate how a functional-related constituent can be linearized to the right, against LP, to the right of the modified head, we must allow for the possibility to delays Phase Projection and Merge:
When a constituent licensed in a functional position is a (nested) complex phase, namely when it bears select features, it would rather be processed in a phase-peripheral position (i.e. on the right). The intuitive motivation of this definition is to reduce complexity (cf. §3.4), by marginalizing nesting. At this point it is unclear whether this is just a preference or a constraint. Following Bianchi and Chesi (2006), I will tentatively assume that the nested phase-head bears some select feature (e.g. a manner phase-head as in (57).a selects a PP as an argument) and, coherently with Phase projection, this feature introduces the required SD at the end of the matrix phase (keeping however its nested relationship with this superordinate phase as in (57).b:

Since [manner] is a functional specification of the verbal phase, it has to be computed while this superordinate phase is still open (thus the adjunct is a genuinely

---

89 A little modification of this definition (i.e. the removal of any reference to the licensed/functional position) would predict heavy NP-shift as well, but this can not be discussed here (see Chesi 2012).

90 Notice that some important constraints (e.g. right-roof constraint) would be readily predictable by (56), as discussed in Chesi (2012).

91 This is necessary in order to prevents the computation of the matrix phase from being interrupted. We have to assume that the top-down projection of this functional specification will follow the top-down projection of the head of the matrix clause.
nested phase, as expected). Only the special (complex) structure of this functional projection is responsible for the right-hand position of the selected PP. Note that the right-hand modifier is not selected by the lexical head (unlike Larson 1988); hence, it is not a sequential phase but a nested one. On these lines, it is possible to capture the unexpected behaviour of some right-hand modifiers which seem to be transparent for extraction (e.g. (51)) by assuming a minimal difference between (57).b, where the island PP is “projected” by the functional specification, and (57).c, where the PP is a sequential phase because the select feature is specified on the verbal head:

(57) c.

\[
\text{[PP } \rightarrow \text{MANNER V]} \quad \text{PP}
\]

Finally, we need to account for the cross-linguistic variation of strong islands (52). An interesting unconventional solution is suggested by Choi & Yoon (2006): while in English predicates select their arguments (P(redicate)-centered language), in Japanese arguments select their predicates (A(rgument)-centered language). The idea of inverse selection is very powerful (and, potentially, very restrictive) but I do not have enough space here to explore many possible implications. Let me simply

---

92 This way the modifier is structurally superior to the VP-internal constituents; this avoids a number of problems with a generalized Larsonian "adjunct as complement" analysis (see Bianchi 2001, among others, for discussion).

93 The idea that a selectional specification for a manner PP can be associated directly to a lexical head is made plausible by the existence of “selected adjuncts” (cf. Rizzi 1990): e.g., a verb like *behave* requires a manner specification; a verb like *weight* requires a measure specification of a certain kind; a verb like *be born* requires a locative or temporal specification, etc.
highlight that in order to accommodate this parameterization within the present model we simply need to allow nominal phases, by means of their case-marking, to project, intersectively, their top-down requirements: that is, a nominative and an accusative nominal phase would conspire so as to project a transitive verbal phase (or rather, the minimal set of verbal shells constituting a sequential-selected verbal phase). From this perspective, case-marked nominals are not strong islands. On the other hand, adjuncts do not select any verbal phase, but they simply license a functional specification of a verbal-phase (in accordance with functional hierarchy discussed in 2.2.4, they lay on the left of the verbal-head). Then we expect adjuncts but not case-marked arguments to behave as nested phases, namely as strong islands.

4.3 Concluding Remarks: Implementing Structure Building Operations
Defining Structure Building Operations means describing an effective way of putting together the building blocks of the linguistic system. Here I assumed these to be part of our linguistic competence: these Structure Building Operations should also be available for any processing task, crucially both for parsing and for generation. Empirical and theoretical details were provided showing that, given a precise definition of parsing and generation, structure building operations such as Merge, Move (Chomsky 1995-2008) and the notion of Phase (Chomsky 1999, 2008) are necessarily part of the grammar. Their effective applicability is however tied to the nature of the process (that, for complexity and explanatory reasons, has been shown to be derivational, top-down, and left-right, Phillips 1996). This “new” perspective required a redefinition of these main components of the grammar: Merge reduced to Lexical Insertion, Movement has been implemented (from left to right) using a Memory buffer, Phases are reinterpreted as complete top-down expectations provided with a single independent (phase by phase) feature-driven memory register. Moreover, these operations have to make an intense use of rich feature structures that have to be encoded within the grammar (cf. Cartographic
Many other empirical phenomena has been analyzed in literature, and evidence has been provided, showing that a better explanation can be found adopting a top-down, left-right Minimalist derivational approach as the one discussed in this book. Notably covert movement and the right roof constraint (e.g. Quantifier Raising in Bianchi & Chesi 2010), rightward movement and its peculiarities, contrasting with the unboundedness of leftward movement (e.g. Extraposition and Heavy NP-shift Chesi 2012), the Leftness Condition on quantificational binding (Bianchi & Chesi 2010), the directional asymmetries in pronoun-antecedent connections (backward vs. forward anaphora) (Bianchi 2009). Many more consequences of this derivational shift need to be assessed yet, but the encouraging results suggest us that we are exploring the right branch of this research tree.
REFERENCES


Choi, Y., Yoon, J. (2006). Argument Cluster Coordination and Constituency Test (Non)-Conflicts, NELS 37, University of Illinois at Urbana-Champaign.


