ABSTRACT

Title of Dissertation: SYNTACTIC IDENTITY AND LOCALITY RESTRICTIONS ON VERBAL ELLIPSIS

Elixabete Murguia, Doctor of Philosophy, 2004

Dissertation directed by: Professor Juan Uriagereka & Professor Amy Weinberg
Department of Linguistics

This dissertation investigates the topic of verbal ellipsis in English. Two main issues are addressed in this work: (i) the identity condition that restricts the application of ellipsis and (ii) the different locality restrictions that apply to elliptical constructions. The identity condition is examined from the point of view of competence, while the locality condition is given a natural answer from the processing domain. Furthermore, a parsing algorithm based on minimalist grammars is defined.

Chapter 1 introduces the topic. Chapter 2 and Chapter 3 deal with the syntactic identity condition. Chapter 2 reviews some proposals in the literature, namely, Lasnik (1995b), Kitagawa (1991) and Fiengo and May (1994). All these analyses examine controversial examples where, apparently, partial syntactic identity between antecedent and gap is found. Chapter 3 presents a new analysis which

assumes late lexical insertion, in the spirit of derivational morphology (Marantz 1993), and offers a unified account of all the cases of partial identity introduced in the previous chapter. It is argued that syntactic identity must be respected, and that the crucial notion for ellipsis is identity of syntactic categories—a condition that is met before lexical items are inserted. Also, the different readings that obtain under ellipsis (i.e., sloppy and strict readings) are explained as emerging at different points in the derivation: before and after lexical insertion, respectively.

Chapter 4 reviews one proposal in the parsing literature (Lappin and McCord 1990) as well as the problems it faces. Chapter 5 offers a processing account of the locality restrictions on gapping (as opposed to VPE and Pseudogapping)), those are analyzed as a result of (i) tense absence/presence (Fodor 1985), (ii) low initial attachment of coordinates, and (iii) Spell-out operations which render syntactic structure unavailable (Uriagereka 1999). A two-fold ellipsis resolution process is presented here—where some work is done on-line, but some at the LF level.

Chapter 6 defines an algorithm based on minimalist grammar operations, precisely on the preference of Merge-over-Move-over-Spell-out (as defined by Weinberg 1999); thus, showing that minimalist grammar models can be translated into computational models. Chapter 7 presents the conclusions.

SYNTACTIC IDENTITY AND LOCALITY RESTRICTIONS ON VERBAL ELLIPSIS

by

Elixabete Murguia

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2004

Advisory Committee:

Professor Juan Uriagereka, Chair
Professor Amy Weinberg, co-Chair
Professor Bonnie Dorr
Professor Colin Philips
Professor Philip Resnik

ACKNOWLEDGEMENTS

There are a lot of people who I am indebted to and to whom I have to express my gratitude, since they have contributed in different ways all these years, either on the professional or on the personal front. I would like to write a few lines here to acknowledge their contributions and to say thanks. I hope that I do not forget to include anyone.

I would like to start by thanking my dissertation director, Juan Uriagereka, who has played a crucial role in making this dissertation possible. I started working with Juan and with the topic of ellipsis for my Generals Paper. Since the beginning of my life as a graduate student, Juan has always been a great source of inspiration not only as a teacher, but also as a director. I am grateful for the time he has very generously devoted to the long and fruitful discussions we have maintained, which helped me clarify my ideas. I am thankful for the challenging questions he has posed, for being always available and willing to discuss syntax, and for his dedication. I would also like to recognize his support and understanding—in difficult times where I needed a friendly and encouraging comment, he was there.

Amy Weinberg, my dissertation co-director, who introduced me to psycholinguistics, also deserves a special mention. I would like to thank her for listening to what I had to say from the beginning, when I was struggling to understand and interpret the ellipsis facts from the view of processing. She patiently listened to what I had to say when not even I was very sure if I was making sense. I would also like to thank her for her pragmatic and practical view on things, something that is also necessary when writing a dissertation.

I cannot forget the rest of the members of my committee. I would like to specially thank Philip Resnik, his contribution to Chapter 6 is enormous. I discovered the world of computational linguistics through different courses that I took with him, and I have to highlight his contagious enthusiasm for his work. I have to thank him for finding the time in his busy schedule to discuss questions and doubts about the algorithm that I was trying to define. He has been of great help. Thanks to Colin Philips for finding the time to discuss and challenge my ideas, as well as for always having further readings to suggest. Finally, I have to acknowledge the work of the external member of the committee: Bonnie Dorr, who has contributed more than what her role as a external member dictates, by posing questions and by patiently pointing out the editing errors.

There are other people that I also have to include here for their contribution on different parts of this work: Howard Lasnik and Norbet Hornstein, for reading previous drafts and for their questions and comments. I would also like to thank those who offered me the chance to present and discuss my work: Myriam Uribe-Etxebarria

and Joseba Lakarra, for their invitation to the conferences organized by the Basque Center for Language (LEHIA); Jon Ortiz de Urbina and Jon Franco, for the presentations both at the Annual Linguistics Colloquium Series, and at the Euskal Mintegia (Universidad de Deusto-Universidad del País Vasco); Marcel den Dikken, for his invitation to the City University of New York Syntax-Psycholinguistics Supper Club Meeting; and Colin Philips for the talk at the CNL Lab Lunch Meeting (University of Maryland). Thanks to the audiences in those events, as well as to the ones who attended The Euresco Conference on Theoretical and Experimental Linguistics and the 16th Annual CUNY Conference on Sentence Processing. Needless to say, the remaining errors are mine.

I would like to express my gratitude to people on the other side of the Atlantic, in particular to Jon Franco, who "dragged" me into Linguistics. Thanks for his help with previous drafts, as well as for supporting me and believing in my potential and my work from the very beginning. Thanks to Jon Ortiz de Urbina too.

To Kathy Faulkingham, our department secretary, who has helped on many different occasions with the boring red tape paperwork. Thanks for always being willing to help.

I would like to say thanks to those friends who have made my years in College Park a richer and more enjoyable experience: (i) the Spanish connection—personified by Leticia Pablos and José—(ii) the Italo-brazilian connection—Luca Massasso, Ana Gouvea, Acrisio Pires and Cilene Rodrigues, and last but not least (iii) the Russian connection represented by Nina Kazanina. Thanks to all of you for your friendship, your support and for the good times and the laughs we had together. ¡Muchas gracias!

There are also friends who left a while ago, but who made my stay in the United States a more pleasant one during the first years of my life as a graduate student: Lucia Quintana, Julien and Christine Mussolino, David Chancogne.

I would finally like to express my immense gratitude to two people whose support and love has been crucial for finishing this dissertation: my friend, Ana Gouvea, and my husband, Mario Bonilla. Thank you Anita for all your support during these past years; thanks for being patient and for listening, for the company and the wonderful and amusing dinners we enjoyed. Thank you Mario for being so patient during hard times, thank you for your understanding and your support, thanks for waiting and also for making life more interesting to live.

I cannot forget to include my mother, who always insisted on the importance of learning, who has supported me in everything from the beginning, and who has very generously given a lot in exchange of nothing.

This work has been financially supported by a scholarship from the Basque Government department of education (Departamento de Educación, Universidades e Investigación) under contract BFI98.218.

TABLE OF CONTENTS

CHAPTER 1: OVERVIEW OF THE DISSERTATION

This dissertation deals with a variety of syntactic constructions that can be grouped under the name of verbal ellipsis. I look at different elliptical structures in English; to be precise, Verb Phrase Ellipsis (henceforth VPE), Gapping, and Pseudogapping. They are exemplified below in (1), (2), and (3) respectively:

(1) a. Mary loves red wine, and Susan does too.

b. He reads the newspaper on the Internet, while she doesn't.

(2) John cooked some paella, and Peter a wonderful pasta.

(3) a. Ann did not excuse his father, but she will her mother.

b. The teacher talked about the problem with the parents after she did with the kid.

The obvious common feature of all elliptical structures is that some elements of the sentence are not present; they have been omitted. In sentence (1), the whole verb phrase in the second conjunct *Susan does too* is omitted. Clearly, the meaning of that sentence is: "Mary loves red wine, and Susan loves red wine", even if the second verb phrase is not overt. In sentence (1), another example of VPE, the verb phrase *read the newspaper* is missing from the subordinate clause. In gapping structures, like (2), the verb is omitted (and optionally some arguments or adjuncts as well); in the case of (2), what Peter was taken to do is to cook a wonderful pasta. In (3), an example of pseudogapping, only the verb *excuse* is omitted, the argument *her mother* is not. In (3) both the verb and the argument *about the problem* are absent.

1.1. Goals of this Dissertation

I will be looking at ellipsis from the point of view of competence, performance, and also from a computational point of view. From the competence side, the questions that I will answer are: (i) How are these sentences built? (ii) What kind of syntactic structure does the elision site have? i.e., are the omitted elements part of the sentence structure or not? And (iii) under which conditions can predicates and arguments be omitted? From the performance side, the issues that arise are relatively similar: (i) How are these sentences assigned structure? How are they parsed? (ii) Do we build syntactic structure for the omitted constituents or not? (iii) If so, how can syntactic structure be computed when there are no overt input items from which to project structure?

Two main issues are addressed in this work: (i) the syntactic identity condition on ellipsis, and (ii) the different locality restrictions that apply for elliptical constructions. It has been argued (e.g. most recently by Chomsky (1993), Lasnik (1995b), Fiengo and May (1994)) that ellipsis is a deletion process that occurs under identity (see (4) below). However, there are examples that seem to argue against this generalization (see (4)). It has also been shown (e.g. Chao (1987), Fodor (1985), and

Berwick and Weinberg (1985)) that gapping as opposed to VPE is subject to locality, i.e. the antecedent needs to be local with respect to the gap (see (5) below):

(4) a. They travel to Europe often, and Peter does ~~travel to Europe often~~ too.

b. They traveled to Europe, and Peter will ~~travel to Europe~~ too.

(5) a. Tom called his friend, and Peter his mother.

b. *Tom called his friend, and I think Peter his mother.

I address the identity condition from the point of view of competence. I propose a minimalist analysis that accounts both for cases of partial syntactic identity (as in (4)), and for the different readings that can be obtained under ellipsis (see example (6) and the strict and sloppy readings in (7) below). I suggest that a more abstract notion of identity is needed, i.e. identity of categories.

(6) John phoned his friend, and Peter did too.

(7) a. John$_i$ saw his$_i$ friend and Peter$_k$ saw his$_i$ friend. (strict)

b. John$_i$ saw his$_i$ friend and Peter$_k$ saw his$_k$ friend. (sloppy)

I address the locality restrictions issue from the point of view of performance. I propose an analysis based on minimalist economy principles that accounts for cases of VPE, pseudogapping and gapping. Locality restrictions for gapping are given a natural account from the processing side, as a result of (i) non-overt tense, (ii) low initial attachment of coordinate structures, and (iii) spell-out operations (in the sense of Uriagereka (1999)) which render syntactic structure unavailable. Verbal ellipsis resolution is described here as a two-fold process which takes place on-line and at LF as well.

From a computational point of view, I define a novel algorithm based on minimalist grammars that accounts for incremental structure building, coordination, as well as VPE, pseudogapping and gapping cases.

1.2 Traditional Accounts of Ellipsis

Traditionally, from the competence side of generative linguistics, there have been two different ways of analyzing ellipsis: the Deletion Hypothesis (e.g. Sag 1976) and the Interpretation Theory (e.g. Williams 1977). I introduce the basic main ideas for each approach here. According to the former, an elliptical sentence like (1) above, repeated here as (8), is base generated as in (9) with a fully realized VP. A deletion rule applies and as a result (9) is obtained:

(8) Mary loves red wine, and Susan does too.

(9) a. Mary loves red wine, and Susan **loves red wine** too.

↓ Rule of deletion

b. Mary loves red wine, and Susan does too.

The application of this rule of deletion is constrained by syntactic identity. Since the first and the second conjunct are structurally identical, deletion may apply and delete the second verb phrase. Syntactic identity guarantees recoverability of deletion, in other words, that the deleted material can be recovered at the interpretive level (i.e. Logical Form (LF)), and that the sentence is assigned the correct interpretation.

The Interpretation Theory, however, claims that a sentence like (8) is base generated with an empty category occupying the position of the second verb phrase, as in (10); this empty category is later on interpreted, giving as a result (10):

(10) a. Mary loves red wine, and Susan does [e] too.

b. Mary loves red wine, and Susan loves red wine too.

There are two problems that deletion theories face: (i) the syntactic identity condition that constrains deletion is questionable in examples where only partial identity seems to be met, at least on the surface (see (11) and (12)); and (ii) identity seems to be operative at levels other than the surface syntactic level (see (13)):

(11) Peter **worked** a lot yesterday, and tomorrow he will ~~**work**~~ too.

(12) I have finished all my readings, and you have ~~finished all **your** readings~~ too.

(13) Paul$_i$ visited his$_i$ friends, and Peter$_k$ did ~~visit **his**$_k$ friends~~ too.

In examples (11) and (12) the elided VP is not identical to antecedent VP. In the first case, the verb in the antecedent is the past form *worked*, while the verb in the elision site is *work*. In the second case, the pronoun in the antecedent clause is the first person pronoun *my*, but in the elision site we find the second person form *your*.

The sentence in (13) exemplifies what has been called a sloppy reading under ellipsis—one in which the pronoun in the elision site is interpreted as referring to the subject in that clause, rather than the subject in the antecedent clause. It shows that identity could be at work not only at the syntactic level, but also at other levels, since the dependency that the elided pronoun establishes inside its own clause has to be parallel to the dependency of the pronoun in the antecedent clause.

The Interpretative approach faces the following problems: if the elided VP is structurally empty, then how are grammatical relations going to be satisfied? How will the subject of the elided clause, for example, receive a theta-role? Also, it has been shown that there are ellipsis sentences in which there is a trace inside the elided VP that needs to be bound (Chao 1987), and that these traces are subject to island constraints (Haik 1987), which argues in favor of having a structured VP:

(14) John knows who$_i$ Bill criticized t$_i$, and Mary knows who$_k$ Sue did t$_k$.

(15) John read everything which$_i$ Bill did t$_i$.

(16) *John read everything which$_i$ Bill believes the claim that he did t$_i$.

In Chapter 3, where the proposal for VPE is introduced, I adopt an approach along the lines of deletion, i.e. I assume that the elided VP is fully structured, and that ellipsis occurs under syntactic identity. Based on the idea of late lexical insertion, I address those cases where sloppy readings obtain (example (13) above), and also those where partial syntactic identity seems to be at work (examples (11) and (12)).[1] I suggest that ellipsis takes place under identity of syntactic categories, and that this condition is met before lexical items are part of the derivation.

1.3 Structure of the Dissertation

Chapter 2 and Chapter 3 of this dissertation deal with ellipsis from the competence point of view. Chapter 2 offers a review of some of the proposals for VPE in the literature; in particular, I concentrate on some proposals related to the issue of the identity restrictions and of the strict and sloppy readings under ellipsis. I discuss Kitagawa (1991) and Fiengo and May (1994), which address the different available readings under ellipsis. Fiengo and May also discuss cases of partial syntactic identity between antecedent and gap. I also look at Lasnik's (1995b) account of the differences in verbal morphology under ellipsis.

Chapter 3 introduces an alternative minimalist proposal for VPE in coordinate structures. Assuming that lexical insertion is a late process in the derivation, I (i) analyze ellipsis as a Null Lexicalization process (rather than deletion or

[1] It has been proposed that lexical insertion is a late process in the derivation, in other words, lexical items are not part of the derivation from the beginning, (Marantz (1993), and Otero (1998)).

interpretation), (ii) account for strict and sloppy readings as emerging at different stages in the derivation of a syntactic structure—if the elided VP is interpreted before lexical items are inserted the sloppy reading obtains; on the contrary, if it is interpreted after lexical insertion, then it is assigned the strict reading—and (iii) reanalyze the identity issue as a condition on syntactic categories—a condition that is met before lexical items are part of the derivation.

Chapter 4 presents a review of one proposal in the parsing literature. Lappin and McCord's (1990) s-structure parsing algorithm for VP anaphora is reviewed. I adopt part of their algorithm for my account of ellipsis in the next chapter. I also present some cases which are not covered by their algorithm which suggest that ellipsis resolution also involves operations at the LF level.

In Chapter 5, I present an account for the processing of the different types of elliptical constructions I have mentioned (i.e. VPE, gapping, and Pseudogapping). A two-fold process in which some work is done on-line, and some at the interpretive level (i.e. Logical Form (LF)) is proposed. Only the minimal syntactic structure is built on-line for the gap—enough structure so as to satisfy grammatical constraints and attach remnants of elision. At the LF level, some interpretation work is carried out—strict and sloppy readings are obtained, as well as Quantifier Raising operations. I assume Weinberg's (1999) algorithm for human sentence processing, and extend it to account for ellipsis and coordination. I offer an explanation for locality restrictions based on (i) the presence/absence of an auxiliary, (ii) low attachment of coordinates, and (iii) spell-out operations.

In Chapter 6, I define a novel algorithm based on minimalist grammars; thus, showing the possibility of creating a computational model based on minimalist principles. I take Weinberg's (1999) human sentence algorithm with the extensions I propose in Chapter 5 for ellipsis and coordination, and translate it into a computational model. I define a non-deterministic parser based on that of Pullman (1986), but modifying the operations proposed there, in order to account for incremental structure building, displaced elements, coordination, and ellipsis. I finish with some conclusions in the last chapter, Chapter 7.

CHAPTER 2: RELATED SYNTACTIC WORK ON VP ELLIPSIS AND PARTIAL SYNTACTIC IDENTITY

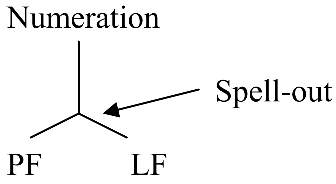
In this chapter I review some proposals that deal with the identity question, i.e. with the condition that ellipsis takes place if syntactic identity between antecedent and gap is respected. All these proposals discuss examples of verbal ellipsis where that condition on identity seems not to be respected; i.e., examples which could argue against a condition on identity for ellipsis. Those proposals include: (i) Lasnik (1995b) who discusses cases of different verbal morphology, (ii) Kitagawa (1991) who deals with examples of strict and sloppy readings, and (iii) Fiengo and May (1994) who investigate both (a) cases of strict and sloppy readings and (b) examples where the arguments in the antecedent verb phrase and in the elision site are syntactically realized in different ways, i.e. examples with different number or gender agreement morphology or even with different nominal expressions (e.g. a pronoun in the antecedent and a reflexive in the elision site or vice versa).

Lasnik (1995b), Kitagawa (1991) and Fiengo and May (1994) show that those examples of superficial sloppy identity (or partial syntactic identity) can be analyzed as examples where syntactic identity is respected, thus, arguing in favor of maintaining the condition of identity for ellipsis.

2.1 Lasnik (1995b): Verbal Morphology: Syntactic Structures Meets the Minimalist Program.

Lasnik (1995) proposes that VPE is the result of a deletion rule that applies at the PF level, before Affix Hopping.[1] His analysis is developed in the minimalist program framework (Chomsky 1993). The minimalist model is depicted in (1):

(1)



A derivation for a sentence starts with a Numeration, i.e. a collection of lexical items, which are assembled into a sentence through subsequent Merge and Move operations.[2] Once the sentence is built, it is spelled-out. In other words, there is a split point where relevant information is sent to the phonetic form (PF) and the logical form (LF) levels.

[1] Lasnik proposes that in languages like French: (i) verbs are fully inflected in the lexicon, and (ii) INFL is a set of strong abstract features that need to be checked—verbs move overtly and check these features. In English, however, it is only with auxiliaries that the situation is parallel to that in French; main verbs (i) are uninflected in the lexicon, and (ii) INFL is an affix that must merge with a bare V. Affix hopping, which attaches the affix to the bare form of the verb, is a morphophonemic rule that applies at PF, and requires adjacency.

[2] The operation *Merge* joins two syntactic elements together, thus, building/projecting syntactic structure—those syntactic elements that are joined can be lexical items or syntactic constituents. The operation *Move* copies syntactic constituents from one position in the sentence to a different one, later on deleting all but one copy in order to satisfy grammatical requirements.

2.1.1 VPE: A Rule of Deletion that Applies at PF under Identity

Lasnik (1995) analyzes VPE as the result of a deletion rule that applies at PF before the rule of Affix hopping, and only when identity of verbal morphology is respected. He shows how syntactic identity between the antecedent and the gap is respected, even in those sentences where the surface string seems to argue against it.

Consider examples (2), (3), (4) and (5) where it seems that tense and aspectual differences between antecedent verb phrase and elided verb phrase can be ignored by ellipsis. There seems to be some kind of sloppy identity (i.e. partial identity) at work here:[3]

(2) John slept, and Mary will ~~sleep~~ too.

(3) John sleeps, and Mary should ~~sleep~~ too.

(4) John was sleeping, and Mary will ~~sleep~~ too.

(5) John has slept, and Mary will ~~sleep~~ too.

Lasnik analyzes all these cases that involve sloppy identity with main verbs as deletion under complete identity by relying on his proposal for verbal morphology in

[3] There are two ways in which the notion of sloppy identity is used in this dissertation, in both cases total identity between antecedent and gap is not met: (i) to refer to cases of partial syntactic identity, where there is a difference in the syntactic realization of certain elements, for example in verbal or agreement morphology between antecedent and gap (see examples (2-5) in text above, (15-17) at the end of section 2.1, and the examples in section 2.3.4 as well), and (ii) to refer to sloppy readings under ellipsis, where for example the pronoun in the elision site is understood as referring to the subject in that clause rather than the subject in the antecedent clause (see examples in section 2.2.1 or 2.3.2). Lasnik uses the term *sloppy identity* only to refer to examples where there is a difference in verbal morphology.

English—i.e. main verbs are retrieved uninflected from the lexicon, and it is through affix hopping that the verbal affix that resides in INFL is attached to the verb.

If the deletion rule applies at PF before affix hopping occurs, then identity of verbal forms holds. Example (2) (repeated below as (6)), consequently, has the structure in (7) when deletion applies:

(6) John slept, and Mary will too.

(7) John INFL **sleep**, and Mary will ~~**sleep**~~ too.

Deletion applies to the structure in (7), which is the structure that the sentence has before affix hopping applies, and which respects identity. The same reasoning applies to the rest of the examples in ((3)-(5)).

This sloppy identity is not found in examples with auxiliary *be*, where total identity must be respected. Consider examples (8), (9), and (10) below:

(8) *John was here, and Mary will ~~be here~~ too.

(9) John will be here, and Mary will ~~be here~~ too.

(10) *The children have been very good here. I wish they would ~~be~~ at home.

In sentence (8), the auxiliary *was* has raised out of the VP in the antecedent, but not in the elided VP, since there it behaves as a main verb. Nevertheless, the reason for its ungrammaticality is not the fact that there is a trace in the antecedent, because (i) a sentence like (11) below where raising has taken place is perfectly grammatical, and (ii) sentence (10) is also ungrammatical and in this case *been* does not raise; the verbal form is different, though. Sentence (9) is acceptable, because the verbal forms in both the antecedent and the gap are identical.

(11) ?John should have left, but Mary shouldn't ~~leave.~~

The case with the verb *have* is similar. If it functions as an auxiliary (examples (11), (12) and (13)) its behavior patterns like auxiliary *be*; total identity must be respected. Instead, when it is the main verb (example (14)), it patterns like the rest of verbs:

(12) *John has left, but Mary shouldn't ~~have left~~.

(13) *The men have left, but the women shouldn't ~~have left.~~

(14) John has a driver’s license, but Mary shouldn’t ~~have a driver’s license.~~

Sentence (11), above, is better than (12), in which the verb forms are different (*has* versus *have*). Even though the forms in the antecedent and the ellipsis site are phonetically identical in sentence (13), the form in the antecedent is the present plural while in the gap it is the base form.

2.1.2 Summary and Comments

Lasnik (1995) deals with one of the problems that any deletion account of VPE has to face, that of differences in verbal morphology between the antecedent and the elision site. Deletion theories defend the idea of verbal ellipsis taking place under syntactic identity. Thus, the examples discussed above by Lasnik are a potential problem for any deletion account since partial identity seems to be at stake. If that was the case, those examples would violate the syntactic identity condition.

Lasnik defines VPE as a deletion process that takes place at the level of Phonological Form (PF) under syntactic identity conditions. He shows how both in the case of the auxiliaries and the main verbs identity of verbal forms is respected. Those examples that involve main verbs with different verbal morphology realizations in the antecedent and the gap are accounted for by assuming that (i) main verbs are uninflected in the lexicon in English, (ii) there is an affix in INFL that must be merged with the bare form of the verb, this happens at the level of PF (Affix Hopping), and (iii) deletion applies before Affix Hopping does, so identity of verbal forms is met.[4]

However, differences in verbal morphology are not the only trigger for sloppy identity in ellipsis examples. Sloppy identity also seems to be at work with gender

[4] It seems plausible to have uninflected verb forms in the lexicon for regular verbs: a form like *decided* is, thus, obtained by merging the affix for the past -*ed* (which resides in INFL) to the bare form *decide* at the level of PF. Nevertheless, it is not so clear how the past form *saw* of the verb *see* can be derived by Affix Hopping. The lexicon seems to be the right repository for irregular forms. If it is so, then the explanation for those examples of ellipsis that involve different forms of irregular verbs in the antecedent is not so straightforward.

and number morphological differences (see (15), and (16)), as well as in cases that involve pronouns in the antecedent and reflexives in the gap (see (17)) or vice versa:

(15) Peter visited his family, and Mary did ~~visit her family~~ too.

(16) The Smiths visited their family, and John did ~~visit his family~~ too.

(17) Tom shaved himself before the barber could ~~shave him~~.

So, the question of how sloppy identity in these cases is accounted for must be addressed. A satisfying answer cannot be that this type of sloppy identity is not important for ellipsis, while identity of verbal forms is. Ideally, we should be able to account for all the cases of sloppy identity in a unified manner.

In Chapter 3, I propose an alternative account of sloppy identity under ellipsis which covers both the cases discussed by Lasnik, where there are differences in the verbal morphology, and the ones in (15)-(17), where differences in gender and number morphology are observed. Thus, mine is an account that unifies all those cases of sloppy identity (where there are morphological differences between antecedent and gap) under the same analysis: identity of syntactic categories.

2.2 Kitagawa (1991): Copying Identity

2.2.1 A Binding Theory (BT) Approach to Ellipsis

According to Kitagawa (1991), VPE is reconstructed and interpreted at the LF level by a VP copy rule. So for a sentence like (18) below, the VP ellipsis will be resolved at the LF level—the VP copying rule applies and copies the antecedent verb phrase into the gap, as represented in (19):

(18) John blamed his son, and Bill did [$_{VP}$ e] too.

(19) LF: John blamed his son, and Bill did [$_{VP}$ blame his son] too.

Kitagawa presents a Binding Theory (BT) approach to VP ellipsis—he argues that Principles A, B, and C of the BT must be satisfied at LF. He suggests that at the LF level the VP copying process (in order to reconstruct the elided verb phrase) and the application of BT need not be extrinsically ordered. Thus, there are two possibilities: (i) the first one is that the VP copy rule applies followed by BT (represented in (20) below); (ii) the second one is that first BT applies and then the VP is copied, represented in (20) below:

(20) a. John$_1$ blamed his$_1$ son, and Bill$_2$ did [blame his$_{2/1}$ son] too.

b. John$_1$ blamed his$_1$ son, and Bill$_2$ did [blame his$_1$ son] too.

If the VP is copied before BT applies, then either the sloppy or strict reading is assigned to the sentence. In the strict reading, the pronoun *his* in the elided clause is interpreted as referring to the subject *John* in the antecedent clause. However, in the case of the sloppy reading, the pronoun is interpreted as referring to *Bill* the subject in the elided clause.[5] If, on the contrary, the VP is copied after BT has applied only the strict reading is obtained (see (21)):

(21) a. Coindexation < VP Copy --- > strict identity

b. VP Copy < Coindexation --- > sloppy identity/strict identity

No matter whether we decide to apply VP copy first and BT after, or BT first and VP copy after, the resulting syntactic object must respect binding theory. Thus, for the sentence in (22) there is only one reading available, the one in (23):[6]

(22) John blamed himself, and Bill did [$_{VP}$ e] too.

(23) a. *John$_1$ blamed himself$_1$, and Bill$_2$ did [blame himself$_1$] too.

b. John$_1$ blamed himself$_1$, and Bill$_2$ did [blame himself$_2$] too.

[5] This is the second sense that the term *sloppy identity* has (see footnote 3 above). In this case, the term refers to sentences where the pronoun(s) in the elided clause hook back to the subject in that clause. In the case of sentence (20) in the text, the different interpretation for the pronoun is the only difference between antecedent and elision site—there are no morphological differences, although there could be morphological differences as well (recall examples (15-17).

[6] This claim will be readdress in the next section, since there are some variations in the readings that different speakers permit. But for the moment let us assume that this is the general pattern.

The reading in (23)—which obtains by coindexation followed by VP copy—is unavailable, since it violates Principle A of the binding theory; the anaphora is not locally bound.[7]

2.2.2 Dialectal Variations

One very interesting observation that Kitagawa makes is that variations in the availability of strict and sloppy readings can be found. He discusses sentences involving the reconstruction of both anaphors and pronouns at LF, and identifies four dialectal variations (Dialect A, B, C and D). We will start by looking at the anaphor examples and then move to the pronouns.

Consider the sentences below, examples involving the copy of anaphors at LF. Binding Theory (Principle A) can be overridden in some cases: for some speakers (Dialect A and B) this happens when there is a gender conflict only, and for others (Dialect C and D) it also can occur when there is no conflict. Kitagawa (1991) analyzes these dialectal variations as the result of the optionality of copying certain features at LF in different dialects:

(24) John considers himself to be intelligent, and Bill does too.

(25) John considers himself to be intelligent, and Mary does too.

[7] Thus, Kitagawa (1991) accounts for the contrast between anaphors and pronouns with respect to strict and sloppy readings by means of the principles of BT, which are independently needed. An independent Variable Rewriting Rule, such as Reflexivization (Williams 1977) or “PRO -> BV” (Sag 1976)—which applies obligatorily to anaphors and optionally to pronouns—is not needed to account for this contrast.

(26) Mary considers herself to be intelligent, and John does too.

Sentence (24), which shows no conflict in gender, has only the sloppy reading for speakers of Dialect A and B—the strict reading is not available since the copying of the anaphor *himself* into the elided verb phrase at LF violates Principle A of the Binding Theory. For speakers of Dialect C and D, however, both the sloppy reading (in (27) below), and the strict reading (in (28) below) are available—speakers of these dialects suppress BT(A) in the output of the copying at LF, and permit the strict interpretation. The feature [+Anaphor] is optionally copied in these dialects:

(27) LF: John$_1$ considers [+Pro, +Anaphor, +Masculine]$_1$, and Bill$_2$ does [$_{VP}$ consider [+Pro, +Anaphor, +Masculine]$_2$ to be intelligent too.
VP Copy < Coindexation

(28) LF: John$_1$ considers [+Pro, +Anaphor, +Masculine]$_1$, and Bill$_2$ does [$_{VP}$ consider [+Pro, ∅, +Masculine]$_1$ to be intelligent too.
Coindexation < VP Copy

Sentence (25), however, has a strict reading for speakers of both Dialect A and B—the conflict in gender between *himself* and *Mary* permits the suppression of Principle A, in other words, the copy of the feature [+Anaphor] is optional in these dialects under the circumstance of gender conflict, as represented in (29) below:

(29) LF: John$_1$ considers [+Pro, +Anaphor, +Masculine]$_1$, and Mary$_2$ does [$_{VP}$ consider [+Pro, ∅, +Masculine]$_1$ to be intelligent too.

For speakers of Dialect B this sentence also has a sloppy reading, accounted for by the optionality of copying the feature [+Masculine], as represented in (30) below. For speakers of Dialect C and D it also has both readings:

(30) LF: John$_1$ considers [+Pro, +Anaphor, +Masculine], and Mary$_2$ does [$_{VP}$ consider [+Pro, +Anaphor, ∅]$_2$ to be intelligent too.

Sentence (26) has a strict reading for speakers of Dialect A, B, C, and D. For those first two, again, BT (A) is suppressed because of gender conflict (the [+Anaphor] feature is not copied. For C and D it is the case that the optionality of copying that feature applies across the board. Only for dialect C there is a sloppy reading for sentence (26) too. For speakers of this dialect the feature [+Feminine] is optionally copied as well.

Consider now sentence (31), (32), and (33) below. For sentence (31) two readings, the strict and sloppy, are available for the speakers of all four dialects. For sentence (32), however, speakers of all dialects except Dialect A—for whom only the strict reading is fine—find both readings acceptable. In the case of sentence (33) it is only speakers of Dialect C that accept the strict and sloppy readings; for the rest of the speakers this sentence is given only the strict interpretation:

(31) John considers his father to be intelligent, and Bill does [$_{VP}$ e] too.

(32) John considers his father to be intelligent, and Mary does [$_{VP}$ e] too.

(33) Mary considers his father to be intelligent, John does [$_{VP}$ e] too.

Once again the dialectal differences are accounted for by the optionality of copying some features. Dialects B, C, and D permit optionality to copy the feature [+Masculine] when there is gender conflict; copying of the feature [+Feminine] is optional in Dialect C too.

In the table that follows below all the different available readings for the four dialects are summarized:[8]

	(24)	(25)	(26)	(31)	(32)	(33)
Dialect A	SL	ST	ST	SL/ST	ST	ST
Dialect B	SL	SL/ST	ST	SL/ST	SL/ST	ST
Dialect C	SL/ST	SL/ST	SL/ST	SL/ST	SL/ST	SL/ST
Dialect D	SL/ST	SL/ST	ST	SL/ST	SL/ST	ST

Table 1

[8] ST stands for strict reading and SL stands for sloppy reading.

2.2.3 Summary and Comments

Kitagawa's proposal offers a good descriptive analysis of the possibilities for sloppy and strict readings under ellipsis in different dialects. He links the availability of those readings to the VP copying rule that applies at LF. This VP copying rule in different dialects allows the exclusion of different features ([+Anaphor], [+Masculine], and/or [+Feminine]).

Two questions should be raised here. First, why should there be a preference to not copy the feature [+Anaphor] over [+Masculine] or [+Feminine]? Abstracting away from dialectal variations, speakers tend to prefer strict readings for sentences (32) and (33) over sloppy readings; in other words, they prefer to interpret a reflexive in the antecedent as a pronoun in the gap rather than interpreting it as a reflexive with a different gender. And second why should the feature [+Masculine] be different from [+Feminine] with respect to its optionality nature? To be precise, why do speakers of Dialect B and Dialect D consider the copying of [+Masculine] optional, but not the copying of [+Feminine]? Why should copying of one gender feature but not the other be optional? What is it about [+Masculine] that makes it special? I give an answer to these questions in the next chapter.

2.3 Fiengo & May (1994)

Fiengo and May (1994) (henceforth F&M) propose that in VPE contexts the elided verb phrase is a reconstruction of the antecedent; the antecedent VP and the elided one are structurally identical. They offer an account for the different readings that are available in ellipsis contexts, i.e. strict and sloppy readings (see example (34) below), as well as for some cases where identity of syntactic structure seems not to be respected.

2.3.1 VPE Eliminates Potential Readings

F&M offer an analysis of the readings that are available in VPE contexts. They observe how ellipsis has an eliminative effect in anaphoric possibilities. Consider examples (34), (36) and (38) below, together with their respective readings (35), (37), and (39):

(34) Max saw his mother, and Oscar did too.

(35) a. Max saw Max's mother, and Oscar saw Oscar's mother. *(sloppy)*

b. Max saw Max's mother, and Oscar saw Max's mother. *(strict)*

Example (35) represents the sloppy reading, in which the pronoun *his* in the elided VP is interpreted as referring to the subject in that clause *Oscar* rather than the subject in the antecedent clause *Max*. In the case of (35), which represents the strict reading, the pronoun in the elision site refers to the subject in the antecedent clause.

(36) Max said he saw his mother, and Oscar did too.

(37) a. Max said Max saw Max’s mother and Oscar said Oscar saw Oscar’s mother. (*across-the-board sloppy*)

b. Max said Max saw Max’s mother and Oscar said Max saw Max’s mother. (*across-the-board strict*)

c. Max said Max saw Max’s mother and Oscar said Oscar saw Max’s mother. (*sloppy-strict*)

d. *Max said Max saw Max’s mother and Oscar said Max saw Oscar’s mother. (*strict-sloppy*)

The across-the-board sloppy reading, where both pronouns in the elision site are interpreted as referring to the subject in that clause *Oscar*, is represented in (37). Example (37) represents the across-the-board strict reading, where both pronouns in the elision site are interpreted as referring to the subject in the antecedent clause *Max*. Example (37) represents a mixed reading, where the first pronoun in the elision site is interpreted as referring to the subject in that clause, and the second pronoun as referring to the subject in the antecedent clause. (37) represents a mixed reading where the first pronoun refers to the subject in the antecedent clause, while the second pronoun to the subject in the elided clause.

(38) Max said he thinks he saw his mother, and Oscar did too.

(39) a. Max said Max thinks Max saw Max's mother, and Oscar said Oscar thinks Oscar saw Oscar's mother. (*across-the-board sloppy*)

b. Max said Max thinks Max saw Max's mother, and Oscar said Max thinks Max saw Max's mother (*across-the-board strict*)

c. Max said Max thinks Max saw Max's mother, and Oscar said Oscar thinks Oscar saw Max's mother. (*sloppy-sloppy-strict*)

d. Max said Max thinks Max saw Max's mother, and Oscar said Oscar thinks Max saw Max's mother. (*sloppy-strict-strict*)

e. * Max said Max thinks Max saw Max's mother, and Oscar said Oscar thinks Max saw Oscar's mother. (*sloppy-strict-sloppy*)

e. * Max said Max thinks Max saw Max's mother, and Oscar said Max thinks Max saw Oscar's mother. (*strict-strict-sloppy*)

f. * Max said Max thinks Max saw Max's mother, and Oscar said Max thinks Oscar saw Oscar's mother. (*strict-sloppy-sloppy*)

g. * Max said Max thinks Max saw Max's mother, and Oscar said Max thinks Oscar saw Max's mother. (*strict-sloppy-strict*)

Those precluded readings in (37) and (39) above are available for the non-elided counterparts of sentences (36) and (38). It is ellipsis that reduces the number of possible readings. F&M account for this fact with their analysis of verbal ellipsis as reconstructing different index types, which are subject to certain identity conditions.

2.3.2 Strict and Sloppy Readings: Reconstruction of α- and β-type Indices

F&M characterize verb phrases under ellipsis as reconstructions, i.e., a set of token structures under an identity condition; or in other words, as occurrences of a sub-phrase marker over a given terminal vocabulary.[9] Reconstructions have the same syntactic structure and the same lexical items. Thus, even though their syntactic structure is identical, phrases like *fly an aeroplane* and *drive a car* are not reconstructions, since they are not built of the same terminal vocabulary.

Reconstructions constitute connected parallel discourse; their role is to introduce fixed points against which to introduce new information. These occurrences can be overt or not; since they are in some sense redundant they do not need to be repeated, ellipsis may apply and omit some of them.

In order for ellipsis to apply, two verb phrases must be reconstructions, in the sense just discussed. Two VPs are reconstructions if (i) the two VPs are built out of the same terminal vocabulary, and (ii) the indexical dependencies present in both VPs are identical. Before we discuss how these two conditions apply with an example, there are two notions that should be introduced: α-occurrences and β-occurrences.

According to F&M (1994), indices are complex objects which have both an indexical value (1, 2… etc), and an indexical type (α or β). α-type indices are independent; they are not licensed through association with other occurrence of the same index. They are evaluated in relation to a context, and reference is established

[9] This is the first definition of the term *reconstruction* that the authors provide in their work. This definition is revised and slightly modified in order to account for examples where there are different nominal expressions in the antecedent and the elided verb phrase. I discuss these cases in section 2.3.4.

independently for each occurrence of the same index. Two α-occurrences are identical if they bear the same index value. However, β-type indices are structurally dependent. They are well-formed if there is another occurrence of the same index value inside the same phrase marker that they can depend on, and get their value from. Thus, they are indexical dependencies (IDs). See (40) where the formal representation of an ID is included:

(40) $<(c_1^{\alpha}, c_2^{\beta} \ldots c_n^{\beta}), I, SD>$

An ID is formed by (i) a sequence of elements (the bearers of the occurrences of index I), (ii) the index value (I), and (iii) the structural description (SD) of the syntactic structure that connects the elements that bear the occurrences of the index. Two IDs are identical, that is i-copies, if they only differ in their index value.

Now, let us see how these notions apply with an example. F&M account for strict and sloppy readings as reconstructions of α-type or β-type occurrences, respectively. Consider (41) below and the two available readings for this example in (42). In (42) an α-type index occurrence is reconstructed, so the strict reading obtains, in (42) a β-type is reconstructed and the sloppy reading obtains instead:

(41) Max saw his mother, and Oscar did too.

(42) a. Max$_1$ [saw his$_1^{\alpha}$ mother] and Oscar$_2$ [saw his$_1^{\alpha}$ mother] *strict*

b. Max$_1$ [saw his$_1^\beta$ mother] and Oscar$_2$ [saw his$_2^\beta$ mother] *sloppy*

VP1-ID < ([$_{NP}$ Max], [$_{NP}$ his]), 1, <NP, V, NP> >

VP2-ID < ([$_{NP}$ Oscar], [$_{NP}$ his]), 2, <NP, V, NP> >

Recall that VPs must obey two conditions to be reconstructions: identical dependencies, and same terminal vocabulary. In (42), the index occurrences in the antecedent and the elided verb phrase are identical; since they have the same value “1”. Also, the terminal vocabulary is the same, so the VPs are reconstructions and ellipsis may apply. In (42), the IDs in both verb phrases are i-copies: they only differ in their index value ("1" in the case of the antecedent, and "2" in the case of the elision), and they are parallel (the pronouns occur in the same context with respect to their antecedents). The terminal vocabulary is also the same, and ellipsis can also apply.

If the parallelism between dependencies is broken, the sloppy reading is not available. Consider example (43) below, where only the strict reading is acceptable. The sloppy reading is precluded because the IDs are not i-copies; they are not structurally parallel. As soon as the parallelism between dependencies is broken, the identity condition is violated, and consequently the verb phrases involved are not reconstructions of each other to which ellipsis may apply:

(43) Max's mother saw him, and Oscar said Mary did too.

(44) a. Max$_1$’s mother [saw him$_1^\alpha$] and Oscar$_2$ said Mary [saw him$_1^\alpha$] *str.*

b. *Max$_1$'s mother [saw him$_1^\beta$] and Oscar$_2$ said Mary [saw him$_2^\beta$] *sl.*

VP1-ID: < ([$_{NP}$ Max], [$_{NP}$ him]), 1, <NP, N, V, NP> >

VP2-ID: < ([$_{NP}$ Oscar], [$_{NP}$ him]), 2, <NP, V, NP, V, NP> >

If the parallelism is restored, the sloppy reading becomes available. The sentence below can have a strict reading, and a sloppy reading, which can be paraphrased as: "Max's mother saw Max, and Oscar said that Peter's mother saw Peter". The pronoun in the elided verb phrase can never be interpreted as referring to *Oscar*, since this interpretation would also break the parallelism of dependencies:

(45) Max's mother saw him, and Oscar said that Peter's mother did too.

2.3.3 Many Pronouns Puzzle and Many Clauses Puzzle

F&M account for the available and the unavailable readings in the Many Pronouns Puzzle—i.e. those examples that involve more than one pronoun in the elision site—with the use of indices too. Consider sentence (36) above, repeated here as (46). The readings are represented under (47):

(46) Max said he saw his mother, and Oscar did too.

(47) a. Max$_1$ said [he$_1^\alpha$ saw his$_1^\alpha$ mother] and (*strict-strict*)

Oscar$_2$ said [he$_1^\alpha$ saw his$_1^\alpha$ mother]

b. Max$_1$ said [he$_1^\beta$ saw his$_1^\beta$ mother] and (*sloppy-sloppy*)

Oscar$_2$ said [he$_2^\beta$ saw his$_2^\beta$ mother]

VP1-ID: < (Max, he, his), 1, <NP, V, NP, V, NP> >

VP2-ID: < (Oscar, he, his), 2, <NP, V, NP, V, NP> >

c. Max$_1$ said [he$_1^\beta$ saw his$_1^\alpha$ mother] and (*sloppy-strict*)

Oscar$_2$ said [he$_2^\beta$ saw his$_1^\alpha$ mother]

VP1-ID: < (Max, he), 1, < NP, V, NP> >

VP2-ID: < (Oscar, he), 2, <NP, V, NP> >

d. *Max$_1$ said [he$_1^\alpha$ saw his$_1^\beta$ mother] and (*strict-sloppy*)

Oscar$_2$ said [he$_1^\alpha$ saw his$_2^\beta$ mother]

VP1-ID: < (he, his), 1, <NP, V, NP> >

VP2-ID: < (Oscar, his), 2, <NP, V, NP, V, NP> >

The readings in (47), (47), and (47) are available because the index occurrences in both VPs are identical. Example (47) involves the reconstruction of an α-type occurrence for both pronouns, the index value is the same in both conjuncts, the VPs are reconstructions and ellipsis can occur. This reconstruction represents the across-the-board strict reading. In (47), which represents the across-the-board sloppy reading, the β-dependencies in the two conjuncts are i-copies—so ellipsis can also occur here. The same reasoning applies to (47), but in this case it is only the first pronoun that is included in the β-dependency copy: the second pronoun is interpreted as strict—but the index value is identical in both VPs.

However, in (47), even though the occurrences for the first pronoun are identical, the ones for the second pronoun are not; the β-occurrences are not in parallel structures; so the SDs are not identical. Thus, the dependencies are not i-copies, the VPs are not reconstructions, and consequently ellipsis is blocked.

The readings for the sentence below ((38) above) are also accounted for in the same way. All the readings in the left column are i-copies to which ellipsis can apply, while those on the right do not respect identity, thus, they are not reconstructions and ellipsis cannot apply.

(48) Max said he thinks he saw his mother, and Oscar did too.

(*strict-strict-strict*)
(*sloppy-sloppy-sloppy*)
(*sloppy-sloppy-strict*)
(*sloppy-strict-strict*)

* (*strict-strict-sloppy*)
* (*strict-sloppy-sloppy*)
* (*strict-sloppy-strict*)
* (*sloppy-strict-sloppy*)

Those examples involving the so-called Many Clause Puzzle (i.e. cases where there is more than one elision site), like (49) below, are accounted for in the same manner. The sentences we have looked at so far had only two clauses: one functioning as antecedent and the other as the gap. However, there can be any number of elided copies of the antecedent phrase; i.e. of reconstructed VPs. In sentences where there is more than one clause elided, the effect of ellipsis on anaphoric possibilities is also eliminative. The availability of readings once again can be justified in terms of the reconstruction of indices that need to respect a condition on identity. See (49) and the available readings in (50):

(49) Max saw his mother, Oscar saw his mother too, but Sam didn't.

(50) a. Max$_1$ [saw his$_1^\alpha$ mother], Oscar$_2$ [saw his$_1^\alpha$ mother], but Sam$_3$ didn't [see his$_1^\alpha$ mother] (*strict-strict-strict*)

b. Max$_1$ [saw his$_1^\beta$ mother], Oscar$_2$ [saw his$_2^\beta$ mother], but Sam [didn't see his$_3^\beta$ mother] (*sloppy-sloppy-sloppy*)

VP1-ID: < (Max, his), 1, <NP, V, NP> >

VP2-ID: < (Oscar, his), 2, <NP, V, NP> >

VP3-ID: < (Sam, his), 3, <NP, V, NP> >

c. *Max$_1$ [saw his$_1^\alpha$ mother], Oscar$_2$ [saw his$_1^\alpha$ mother], but Sam$_3$ [saw his$_3^\beta$ mother] (*strict-strict-sloppy*)

VP3-ID: *<(Sam, his), 3, <NP, V, NP>>

d. *Max$_1$ [saw his$_1^\beta$ mother], Oscar$_2$ [saw his$_2^\beta$ mother], but Sam$_3$ didn't [see his$_1^\alpha$ mother]. (*sloppy-sloppy-strict*)

*his$_1^\alpha$

Sentence (49) only permits across-the-board strict or sloppy readings, as in (50) and (50) respectively. No mixed readings in which one elided clause has a strict reading and the other a sloppy reading (50), or vice versa (50), are allowed. The reason for this is that those readings involve reconstructing an α-occurrence as a β-occurrence (or vice versa), violating identity of indices. Thus, the verb phrases are not reconstructions, and ellipsis cannot apply.

2.3.4 Vehicle Change

F&M study some examples that involve sloppy syntactic identity between the antecedent and the elided VP, i.e., where the syntactic form of an argument is altered among the tokens of the reconstruction.[10] In order to account for these cases, while maintaining the characterization of elided verb phrases as reconstructions of their antecedents, they propose an operation called Vehicle Change which allows for free adjustment of feature values. See examples below:

(51) Peter blamed himself before his boss did.

(52) Peter blamed **himself** before his boss [blamed **him**].

(53) John visits his family often and Mary does too.

(54) John visits **his** family often, and Mary [visits **her** family often] too.

[10] I have already discussed some examples of sloppy identity in the form of distinct verbal morphology between the antecedent and the elided clause (section 2.1.1). I have also talked about sloppy identity in the form of sloppy readings (sections 2.3.1, 2.3.2, and 2.3.3). In this section, I discuss sloppy identity in the syntactic realization of arguments. Some of the cases presented here have been introduced in section 2.2.1, in our discussion of Kitagawa´s work.

In sentence (51) the operation of Vehicle Change allows the reflexive in the antecedent to be reconstructed as a pronoun, at least under the sloppy reading in (52). The case with sentence (53) is similar; vehicle change allows the masculine pronoun in the antecedent to be realized as a feminine pronoun in the gap, as represented in the sloppy reading in (54).

Reconstruction requires that each token structure have the same set of arguments; non-arguments and non-argumental parts of NPs are not relevant. In other words, there are certain aspects of syntactic structure that cannot be distinguished by reconstruction. So, some elements in the reconstructed structures might be realized differently, and still be non-distinct.

There are two types of vehicle change: (i) Type A, when one sort of expression is realized in various ways (e.g. nondistinctiveness of pronouns, reflexives, and PRO; or nondistinctiveness between a name and a variable), and (ii) Type B, which refers to cases of indescernibility of values within a syntactic feature paradigm (e.g. agreement features).

Let us start by considering Type A. F&M (1994) claim that reflexives are not atomic elements, but they are rather composed of two parts: the pronominal part (the argument part) and the grammatical formative *self*, which triggers Principle A of the Binding Theory. Pronouns and reflexives are non-distinct for reconstruction, since in order to respect identity, all that is required is that the argument part is reconstructed. Thus, the two expressions in (55) below are non-distinct. Consider examples (56) and (58):

(55) $[_{NP}[_{NP}\text{him}]\text{self}]$ $[_{NP}\text{him}]$

(56) Max hit himself before Oscar did.

(57) a. $\text{Max}_1{}^{\alpha}$ hit $\text{him}_1{}^{\beta}$+self before $\text{Oscar}_2{}^{\alpha}$ [hit $\text{him}_2{}^{\beta}$+self].

b. $\text{Max}_1{}^{\alpha}$ hit $\text{him}_1{}^{\beta}$+self before Oscar [hit $\text{him}_1{}^{\beta}$].

(58) Bush voted for himself, and Barbara did.

(59) a. $\text{Bush}_1{}^{\alpha}$ voted for $\text{him}_1{}^{\beta}$+self, and $\text{Barbara}_2{}^{\alpha}$ [voted for $\text{her}_2{}^{\beta}$+self].

b. $\text{Bush}_1{}^{\alpha}$ voted for $\text{him}_1{}^{\alpha}$+self, and Barbara [voted for $\text{him}_1{}^{\alpha}$].

Both examples are acceptable under the strict and the sloppy readings.[11] For reconstruction, a nominal can take any syntactic form as long as its indexical type and value is unchanged. For example, sentence (56) under its strict reading (in (57)) respects identity, since even though only the pronominal argument part of the reflexive has been reconstructed, the indexical type and value did not change. The strict reading for sentence (58), under (59), can be accounted for in the same way. Example (60) below shows that reconstruction is a symmetrical relation; therefore, in the same way that *himself* can be reconstructed as *him*, the opposite also holds, i.e. *him* can be reconstructed as *himself*:

(60) Barbara voted for him, but Bush didn't.

[11] Whether an α-reflexive or a β-reflexive is reconstructed depends on the context, it is a result of the interaction of Binding Theory (Principle A, B and C), and Dependency Theory (α or β index types).

(61) Barbara$_1{}^{\alpha}$ voted for him$_2{}^{\alpha}$, but Bush$_2{}^{\alpha}$ didn't [vote for him$_2{}^{\alpha}$+self].

Pronouns and reflexives are nondistinct for reconstruction; the important thing is that an expression of type [+pronoun] is reconstructed—which syntactic form it adopts is a function of the context in which it occurs. [12]

Consider (62) now, an example of the non-distinctiveness of names and variables. Names and variables are the lexical and the null forms, respectively, of a cell of the nominal typology (like pronouns and PRO are). Identity holds as long as the indices in both do respect identity conditions.

(62) John kissed Mary, but I wonder who Harry will.

(63) John kissed Mary$_1{}^{\alpha}$, but I wonder who Harry [kissed e$_1{}^{\alpha}$].

Type B of vehicle change (featural vehicle change) refers to cases where there is indescernibility within syntactic feature paradigms. Example (64) involves agreement features, and example (65) the *some/any* alternation:

(64) I turned in **my** assignment, but most of the students didn't [turn in **their** assignments].

(65) Max didn't talk to **anyone**, but Oscar did [talk to **someone**].

[12] F&M (1994) propose the existence of only one anaphoric feature [pronoun]. Reflexives and pronouns are [+pronoun], and referring expressions such as names are [-pronoun]. The feature [anaphor] does not exist in the nominal typology they propose.

This sort of featural vehicle change interacts with Binding theory. Example (66) illustrates this: the feature [+/-pronoun], which distinguishes pronominal from nonpronominal elements, is affected by vehicle change here. Consider the reading in which the pronoun *he* is understood as anaphoric to *John.* Then if the name *John* is reconstructed in the elided site, Principle C of the BT will be violated. If instead vehicle change applies, and the feature [-pronoun] is switched to [+pronoun] then the reconstruction conforms to BT requirements. Here, vehicle change reconstructs what F&M call the pronominal correlate, which is subject to Principle B of BT.[13]

(66) Mary loves John, and he thinks that Sally does too.

(67) Mary loves John_1^{α}, and he_1^{α} thinks that Sally loves $^{P}\text{John}_1^{\alpha}$

Featural (Type B) and non-featural (Type A) vehicle change can interact, as example (68) shows. In this case, the reflexive form of the pronominal correlate of *John* is reconstructed; this avoids violating Principle B:

(68) I shaved John, because he wouldn't.

(69) I shaved John, because he wouldn't shave ^{P}John+self.

[13] A is a pronominal correlate (P) of B iff where B has [-pronoun] A has [+pronoun]. This must be the only difference.

F&M also include examples of nonpronominal vehicle change, i.e., featural vehicle change where a pronominal element in the antecedent is reconstructed as a nonpronominal element in the elision, an example is (70):[14]

(70) Which paper$_1$ did the student who was supposed to read it$_1{}^{\alpha}$ refuse to [read ${}^{p-}$e$_1{}^{\alpha}$]?

Pronominal correlation from a variable to a pronoun can also be found, as example (70) shows. In the following example, if what is reconstructed is the verb and the trace of the wh-element, the resulting structure will be ungrammatical since the trace will not be bound. Instead, if the pronominal correlate of the variable is reconstructed the sentence is acceptable:

(71) John named a country which$_1$ he wants to visit e$_1{}^{\alpha}$, and given the amount of traveling he does, I am sure that he will [visit ^{P}e$_1{}^{\alpha}$].

Let us finish the discussion of examples that involve sloppy identity with the sentence in (72) below, where the antecedent is in passive voice, and the elision site in active voice:

[14] They observe how nonpronominal vehicle change, where a [+pronoun] in the antecedent is vehicle changed into a [-pronoun], is practically unobservable, since all the violations of Principle B are included in Principle C.

(72) This law restricting free speech$_{\alpha}^{1}$ should be repealed by Congress e$_{\alpha}^{1}$, but I am sure that it won't [repeal this law restricting free speech$_{\alpha}^{1}$].

F&M propose that arguments are composite elements, they consist of two parts: (i) an argument expression, and (ii) an argument position. Neither of these constitutes an argument in and of itself, and they may appear in the same position or they might occupy two different positions, i.e. they form a chain. The argument *this law restricting free speech* in the first clause in sentence (72) forms a chain. In the second clause, however, that same argument is not spread out in the sentence. Nevertheless, those two VPs are reconstructions. Even though the VP argument in the antecedent is realized as a trace, the two VPs are reconstructions, since both are occurrences of the same verb and argument.

Thus, in order to account for all the cases of partial identity in this last section, F&M redefine the concept of reconstruction as a set of token structures over a given terminal vocabulary of predicates and arguments.

2.3.5 Summary and Comments

F&M (1994) present an analysis of verb phrase ellipsis that defends the idea of the elided VP being structurally identical to the antecedent. The elision site is categorically fully specified, though there is no lexical material. They define elided VPs as reconstructions of their antecedents, i.e. as occurrences of the same sub-phrase marker—where identity of lexical items must be respected by the predicates and the arguments (at least by the argument part).

Their proposal accounts for the anaphoric possibilities, and for both the available and the unavailable readings under ellipsis by proposing the existence of two different index types. Strict readings are proposed to involve the reconstruction of an α-type index, and sloppy that of a β-type index.

F&M also deal with the problem of apparent sloppy identity, or partial syntactic identity. In their case, they study nominal expressions that are syntactically realized in different ways among the occurrences of the VP (as Lasnik (1995b) above did with respect to differences in verbal morphology). They show that these differences can be accounted for with an operation called Vehicle Change, which allows maintaining the description of verb phrases under ellipsis as reconstructions.

In the next chapter, I offer an alternative analysis of VP ellipsis, which accounts for all the different cases of sloppy identity presented in this chapter. This new proposal is based on the Minimalist Program (Chomsky 1995), it does not make use of different index types, and it offers a unified account for all the cases of sloppy identity (i.e. partial syntactic identity) and of sloppy readings. It is proposed that, by assuming late lexical insertion, strict and sloppy readings are obtained in a derivational way, as a result of the stage at which the elided VP is interpreted. The problem raised by differences in verbal morphology and differences in nominal expressions is also addressed there. F&M discuss differences between nominal expressions in their work, but not between verbs.

2.4 Conclusions

We have seen with the discussion of the proposals above that sloppy identity (or partial syntactic identity) manifests itself in different fashions: in the form of different verbal morphology (as discussed by Lasnik (1995b)), in the form of sloppy readings (as discussed by Kitagawa (1991) and F&M (1994)), and in the form of different gender and number morphological realizations, or even as different anaphoric expressions (as discussed by F&M (1994)).

We have also seen that all these cases of partial syntactic identity can be accounted for, and consequently, that the condition of syntactic identity that constraints the application of ellipsis under deletion type theories can still be maintained if it is relaxed (like in the case of Vehicle Change for example). Ellipsis occurs under syntactic identity.

In the next chapter, I offer a unified treatment of all these cases of partial syntactic identity. I propose that the condition that constrains the application of ellipsis is identity of syntactic categories.

CHAPTER 3: A MINIMALIST ACCOUNT OF VP ELLIPSIS

In this chapter, I introduce an account for VPE that is developed inside the Minimalist Program framework (Chomsky 1993, 1995). Coordinate elliptical structures are analyzed, together with the available readings in those contexts. The proposal discussed here is based on that of F&M (1994), which defines verb phrases in elliptical constructions as reconstructions that are syntactically fully realized, even though there might be no lexical material inserted. As we saw in the previous chapter (section 2.3.2), F&M view strict and sloppy readings as a reconstruction of different index types—the so-called α- and β-occurrences. But the approach taken here is derivational rather than representational; in other words, strict and sloppy readings are explained as a result of interpreting the elided VP at different stages in the derivation, instead of by means of using representational devices such as index types.

I assume that lexical insertion is a late process in the derivation along the lines of Distributed Morphology (Marantz 1993). By adopting this view on lexical insertion, ellipsis can be accounted for as a null lexicalization process (i.e. no lexical items are inserted in the elision site) —rather than deletion or interpretation—and a derivational account for strict and sloppy readings can be advanced. Depending on which stage of the derivation the elided verb phrase is interpreted at—i.e. prior to or after lexical insertion—the sloppy or the strict readings are assigned to the elided structure.

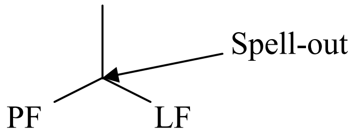
The issue of the syntactic identity condition between antecedent and gap that must be met in order for ellipsis to apply is also discussed here. A more abstract notion of identity is proposed: identity of syntactic categories, which is met before lexical items are inserted in the derivation. With this new notion of identity, cases of the so-called sloppy identity can be explained; to be precise, I deal with cases of (i) verbal morphology differences, (ii) agreement differences (e.g. gender), and (iii) pronoun/reflexive (or vice versa) differences.

3.1 Some Theoretical Background

3.1.1 A very Brief Introduction to the Minimalist Program (Chomsky 1993, 1995)

The new proposal for VPE that is about to be introduced is developed inside a minimalist framework (Chomsky 1993, 1995). The language model that this syntactic theory develops is depicted in (1). For a sentence like *The woman found the place* the derivation will start with the following elements in the Numeration:

(1) Numeration: {the$_2$, woman$_1$, found$_1$, place$_1$}



The derivation of any sentence starts with a Numeration, which can be defined as a collection of lexical items and their syntactic features: every lexical item in the numeration is a token, since it is the instantiation of a type. Each lexical type has a sub-index attached that indicates how many tokens of it can be pulled out of the numeration to build the syntactic structure of a sentence.

Words are pulled out of the Numeration and combined by way of an operation called *Merge*—which joins two syntactic objects together (see definition of the term *syntactic object* in footnote 2 of the current chapter) and projects syntactic structure. Every time that *Merge* pulls out a word from the Numeration its sub-index is modified (e.g. once the word *woman* has been used once the sub-index it holds in the numeration will be modified to zero, indicating that this word/token is no longer available in the Numeration for future *merge* operations).

The second operation that builds syntactic structure is *Move* which copies syntactic constituents form one position in the sentence to a different one, thereby deleting all copies but one (hence the "move" metaphor). The reason for movement operations is always syntactic features that need to be checked at certain positions in the tree.

Once the Numeration has been exhausted—in other words, when there are no tokens left in the numeration—and all relevant syntactic features have been checked through *merge* and *move* operations, then Spell-out applies. This last operation splits the derivation and sends only the relevant information to both the Phonetic Form (PF) level and the Logical Form (LF) level. Thus, information about the pronunciation of a sentence is sent to the PF level; while information about the interpretation of a sentence is sent to the LF level.

The previous paragraphs summarize the main ideas inside the minimalist theoretical framework. However, it has been proposed by Uriagereka (1999) that there can be more than one Spell-out operation in a derivation, and that certain properties can be deduced from this assumption. I assume this view on Spell-out for my work and I discuss Uriagareka´s proposal as well as its theoretical implications in the next section.

3.1.2 Multiple Spell-Out (MSO) (Uriagereka 1999)

MSO is an attempt to reduce Kayne's Linear Correpondance Axiom (LCA) to a more minimalist basis.[1] According to Uriagereka (following Chomsky 1995:chapter 4), the linearization condition follows from bare output considerations—the phrase marker has to be linearized, otherwise it will be an ill-formed object at PF. The LCA is deducible from economy considerations, under Uriagereka's view.

The first concept we have to address is a Command Unit (henceforth CU), which Uriagereka defines as an object that emerges through the monotonic application of merge, as in (2):

[1] Linear Correspondance Axiom (LCA)
Base Step: If @ commands &, then @ precedes &.
Induction Step: If $ precedes & and $ dominates @, then @ precedes &.

(2) a. Command Unit

{@, {$, {@, {@, {&…}}}}}

Merge
↑
$ ←-→ {@, {@, {&…}}}

Merge
↑
@ ←-→ {&…}

(Monotonic application of merge to same object)

b. Not a command unit

{@, {{$, {$, {%…}}}, {@, {@, {&…}}}}}

Merge
↑
{$, {$, {%…}}} ← → {@, {@, {&…}}}

Merge *Merge*
↑ ↑
$← → {%…} @ ← → {&…}

(Non-monotonic application of merge to two separately assembled objects)

It is within CUs that syntactic terms communicate. The order resulting from linearizing each CU is based on the command relation and on the history of merge. For Uriagereka, following Epstein, command is a reflex of merge, and the head-complement merger codes the basic PF order between them. Command maps to precedence in simple CUs, because this is the optimal state of affairs. The function mapping command to precedence is less costly information theory-wise. Thus, he deduces Kayne's LCA base step (object in (2)).

According to Uriagereka (1999), after abandoning the D- and S-structure levels, there is no point on restricting Spell-out to one application, because that is only a residue of the formerly existing S-structure level. He presents a dynamically split model, in which multiple application of Spell-out happens, accessing PF and LF in separate derivational cascades. There are structures that involve more than one CU (e.g. the object in (2)), and which are not linearizable if we do not adopt such a system. By assuming a dynamically split access to interpretation, the system satisfies the Induction step of the LCA.

Uriagereka (1999) argues that CUs are spelled-out separately since that is the most economical alternative. Spell-out separates phonetic from categorial/semantic features, yielding structures that are interpretable by the PF and LF components. After Spell-out, what remains is not a phrase marker any longer, it is in effect a lexical compound. This element is frozen, so the syntax cannot operate with it any longer, but it can associate further up. It is not a syntactic object, but it has a label and terms, which are interpretable.[2, 3]

[2] A syntactic object is defined by Chomsky (1995): chapter 4, in the following manner:
Base: A word is a syntactic object.
Induction: {@, {L, K}} is a syntactic object, for L and K syntactic objects and @ a label.

[3] According to Chomsky (1995):
Label: Within a syntactic object, a label @ is not a term.
Term: K is a term if and only if (a) or (b):
Base: K is a phrase marker.
Induction; K is a member of a member of a term.

Here is how the Induction step can be derived, according to Uriagereka: "The elements $ dominates (in (2)) should act as $ does within its command unit, this is a consequence of the fact that $ has been spelled out separately from the CU it is attached to, in a different derivational cascade. These elements cannot interact with those elements that $ does, in the mother CU, their place in the structure is frozen under $'s dominance."

I assume this MSO system for our analysis of VPE, but it should be said that the function that multiple spell-out has in the system that follows is going to be limited to the mere job of supplying an order to the syntactic structure. As we will see below, there are occasions when MSO applies in the derivation that lexical items may have not been inserted yet. And consequently, phonological and semantic features cannot be shipped to the PF and LF components.

3.2 A Derivational Account of VPE and of Strict and Sloppy Readings

F&M (1994) define VPs in VPE as reconstructions (see Chapter 2, section 2.3.1), where the elided VP is categorially fully specified, lacking only the phonetic material. I share this view with them. However, they do not say anything about how the different occurrences of the VP and the coordinate structures are built or linearized, or about the kind of process that is responsible for elision. Is it deletion or interpretation? In this section, I offer some suggestions with respect to those issues. Elision is analyzed as null lexicalization. This is possible, because in the model presented here lexical insertion takes place late in the derivation, and because the structural parallelism that is needed for elision to apply comes for free from the iteration rule that builds coordinates. We see this in the next two sections.

3.2.1 Delayed Lexical Insertion

It has been proposed in the literature, most recently by Distributed Morphology (Marantz 1993) or (Otero 1998), that lexical insertion is a late process in the derivation, and that there are some linguistic facts that could not be accounted for otherwise.

I adopt this view and argue that there is a division between what is purely formal and the substantive lexicon, and also that lexical items are not inserted when the derivation starts, but later on. This idea of late lexical insertion is going to be the basis for my analysis of ellipsis and of the readings under ellipsis.[4]

Thus, the Numeration cannot be defined as a collection of lexical items any longer, but rather as a collection of syntactic categories and features with which the derivation starts. The Numeration is restricted to formal elements. Lexical insertion

[4] One could ask whether having syntactic structure built out of categories—to only later on insert lexical material—is taking one step back to the way syntactic derivations were thought of at the time rewriting rules, such as the ones below, were considered to build sentences:

(i) NP → Det N
Det → the

Nowadays, there are two approaches with respect to the way derivations are carried out: (i) lexically driven derivations and (ii) derivations of the sort proposed by Distributed Morphology (Marantz 1993), which defend the idea of late lexical insertion. Both are possible approaches and still under consideration, so we are not adopting an old view which has already been rejected.

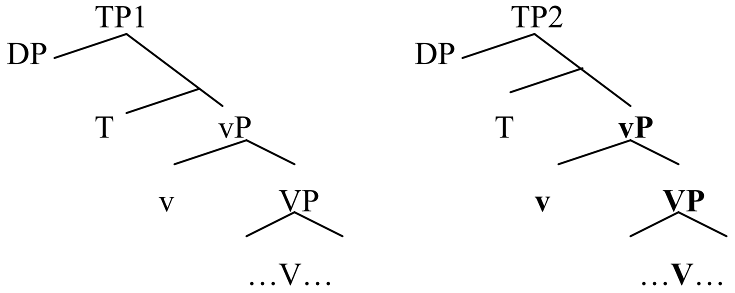
takes place only when the numeration has been exhausted, all the elements merged and the pertinent constituents moved to check its features. Lexical information—that is, the phonological and semantic information associated with each word—does not come into play until the very last minute.

3.2.2 A Rule of Iteration

In this section I want to show how the elided VPs in coordination can be constructed by the application of an iteration rule that affects part of the syntactic structure (the VP). This iteration rule gives us for free the syntactic parallelism that has been argued to be needed, in order for ellipsis to apply.[5] Syntactic parallelism is a natural consequence, since the same syntactic structure is built more than once. For a sentence like (3), the rule of iteration builds a second VP (in (4), in bold), which is structurally identical to the antecedent:

(3) Tom likes movies, and Peter does too.

(4)



[5] It has been argued in the literature (most recently by Chomsky 1993; 1995) that for ellipsis to apply a condition on structural parallelism has to be met first.

F&M (1994) argued that VPs in VP ellipsis are reconstructions, literally carbon-copies, or a set of token structures over a terminal vocabulary. These different occurrences of the same sub-phrase marker constitute connected parallel discourse and are redundant—that is the reason why ellipsis may (though need not) apply. I analyze these VPs which are carbon copies as the result of this iteration rule that rebuilds the structure of the antecedent VP for the elision site.

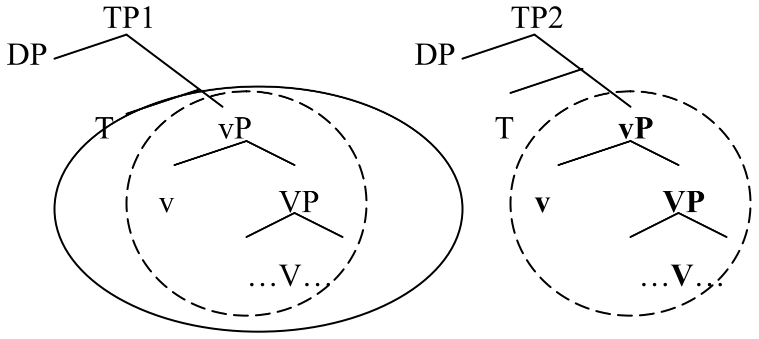
3.2.2.1 Tokens, Types, Token Structures and Occurrences.

Before we continue with the discussion, I would like to include a brief comment on some terminology used here. The terms that are to be defined beforehand include the following: type, token, occurrence, token structure and occurrence of a token structure (or of a sub-phrase marker). I have briefly characterized the terms *type* and *token* in the introduction section (3.1.1 above). There, we said that examples of the former are *woman, see, at*, etc. and of the latter $woman_2$, see_1, at_1, etc; or in other words every time a type is instantiated it becomes a token. In the proposal for VP ellipsis presented here, since lexical insertion takes place late in the derivation the term *token* refers to every instantiation of a type (i.e. V, N, D, etc.) in the Numeration (i.e. N_1, V_1, D_2). In other words, a token is an instantiation of a syntactic category with its syntactic features in the numeration. The sub-index on every token indicates the number of times that token is pulled out from the numeration when building the syntactic structure of a sentence.

Let us consider now the concept of *token structure*, and see how the concepts of token and type apply to the case of syntactic structures. In the case of syntactic structures, a type is for example a VP or an NP; once we have an instantiation of one of these types, i.e. one concrete example of a VP or an NP, we have a token structure. Let us see how these definitions apply with a concrete example. Consider sentence (3), repeated below in (5)—an example of verbal ellipsis (the elided VP is between brackets)—and its sentence syntactic structure in (6):

(5) Tom likes movies and Peter does [like movies] too.

(6)



In (6) above, there is an example of a token structure: the vP that is surrounded by an oval continuous shape; the antecedent vP. This token structure is built with tokens from the Numeration.

However, there are two occurrences of this token structure in that example (inside dashed circles): (i) one built with the tokens from the numeration (the antecedent VP), and (ii) one built by the iteration rule (the elided VP). Both VP occurrences are built from one unique token in the Numeration (see Numeration for this sentence in (9) below): one V token and one N token. If these two occurrences of the same token structure need to be lexicalized, they will be lexicalized in the same way, for example the verb cannot be lexicalized as *like* in the antecedent and *watch* in the elision, since they are built out of the same tokens in the numeration (I will come back to the issue of tokens, occurrences and identity in sections 3.2.4.1 and 3.3 below).[6]

The iteration rule in the example above has only built one occurrence of the token structure (of the elided vP). But more occurrences of that token structure can be built by this iteration rule, as in sentence (7), where there are two elided VP and therefore two occurrences of the vP built by that iteration rule:

(7) Tom likes movies, Peter does too, but Mary doesn´t.

3.2.2.2 The Derivation of one VPE Example: Iteration and Null Lexicalization

After defining the terms of *token structure* and *occurrences of a token structure* we can continue with the discussion of the proposal. I share F&M´s idea of VPs under ellipsis as occurrences of the same syntactic structure, and I analyze these as the result of iteration. In the system proposed here, the VPs are a set of token structures

[6] Another example of different occurrences of a token structure is the case of syntactic movement. In the case of movement—if this is viewed along the lines of the copy theory of movement (Nunes 1995) where an identical copy of the moved constituent is left behind, rather than a just a trace—we could also say that there are two or more occurrences of a token structure.

The constituent which moves from its base position is the token structure that is built with tokens from the numeration. This constituent in its base position is one of the occurrences of that token structure. Together with this occurrence, there will be as many occurrences of the same token structure as landing sites for the movement operation.

over categories, rather than over a terminal vocabulary (since lexical insertion has not taken place at this stage that the iteration does).

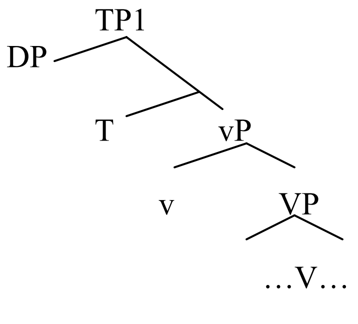
Having said that, let us see how all the notions I have just discussed apply with one example. According to Goodall (1987), coordinators subcategorize for one or more element, so every time that we have a coordinator as part of our numeration, we need two or more occurrences of a sub-phrase marker. Consider the sentence in (8), and the numeration for that sentence, which will be something along the lines of (9):

(8) Tom likes movies and Peter does too.

(9) {D[-pronominal]$_1$, D[-pronominal]$_1$ T[present]$_2$, V$_1$, N$_1$, B$_1$}

The derivation starts with these categories. The first syntactic object that is going to be assembled is the TP1 in (10) through subsequent merge and move operations, and the numeration is reduced to the elements in (11):

(10)



(11) {D[-pronominal]$_0$, D[-pronominal]$_1$, T[present]$_1$, V$_0$, N$_0$, B$_1$}

Since coordinative or Boolean elements—such as *and*, *or*, *but*—subcategorize for two or more elements, the iteration rule rebuilds part of the phrase marker in a parallel plane (Goodall 1987).[7]

The iteration rule applies in the case at stake to the VP (or vP) sub-phrase marker.[8] This iteration rule builds a second vP. This iterated VP is merged with the rest of the elements in the numeration, to end up with two parallel syntactic structures, as in (12):

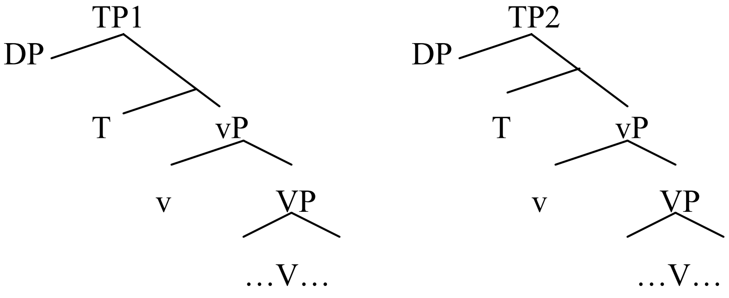
[7] We assume Goodall (1987) which offers an alternative analysis of coordinate structures as parallel structures. Conjunctions subcategorize for two or more phrase markers that exist in parallel planes: syntactic operations must apply in parallel too. He argues that not all phrase markers can be represented as trees, and he identifies coordination to be one of those cases. Coordinates can be represented as phrase markers with more than one string of terminals.

[8] In languages like Spanish, ellipsis affects more structure; it deletes the whole TP, and the iteration rule works with all that material. It, thus, appears that the scope of the iteration rule is parametrizable, as well as the amount of structure that null lexicalization can affect.

(ii) Maria ama a su madre enormemente pero Juan no.
Maria love-3p.sing her mother enormously but Juan not
"Maria loves her mother enormously, but Juan doesn't"

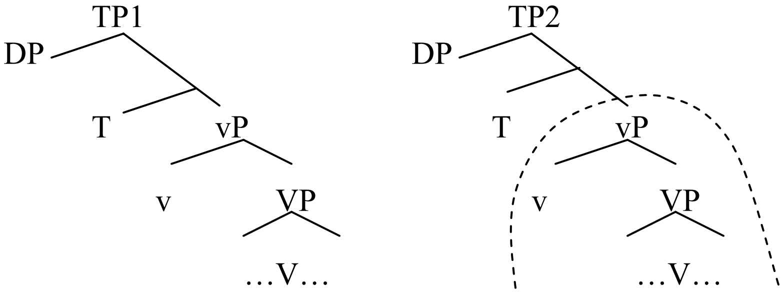
This difference in the amount of material that is deleted may be due to (i) the difference in verb raising that exist among languages—in languages like English where there is poor morphology the verb does not raise to T, while in languages with rich morphology like Spanish the main verbs raise to T—or (ii) to the difference in the position of Sigma Projection (ΣP) (the functional category where elements like negation reside) (Laka 1990)—in English ΣP is below TP, while in Spanish ΣP is above TP, and ellipsis affects that material that is below ΣP. Whether what is triggering the difference is (i) or (ii), or a combination of both, is not important. The result is that different amount of material is deleted in both languages. Martins (1994) offers an account for ellipsis in Portuguese based on the notion of ΣP. Lopez (1997) analyzes the differences between English and Spanish in terms of ΣP and the auxiliary. Murguía (1997) also looks at the differences in terms of the different position that ΣP occupies.

(12)



This iteration rule builds k-occurrences of the same token structure ("carbon-copies" in F&M's terms) which constitute connected parallel discourse. Since these iterated structures are redundant, null lexicalization may apply. That is, when lexical items are inserted in the derivation, those categories that are part of the iterated VP (under the dashed arc in (13)) will not need to be lexicalized:

(13)



The iterated VP is the same object as the first built VP, it is has identical syntactic structure, and thus, it is redundant material. What triggers this process of null lexicalization is the structural parallelism found between the first and the second VPs; they are built out of the same syntactic categories, they are identical.

Besides syntactic information, lexical items provide two types of information: phonetic and semantic (π, λ). This means that those categories that are not lexicalized (the categories that build the iterated VP) will not be pronounced at PF, and at LF they will have no semantic features associated. So, the VP that is lexicalized (the overt VP) will function as the antecedent when interpreting the elision, and substantive representations (lexical items) will be copied from the antecedent VP into the elided VP (we will see this with more detail in section 3.2.4.1 and 3.3). The fact that both VPs are built from one token in the numeration ({V_1, $N_{1...}$}, see numeration in (9)) guarantees that the different occurrences of that token structure will contain the same lexical material.

In a sentence like (14) more elements than those which are phonetically realized are part of the structure. Although no overt lexical material is present in the gap, the elided VP is structurally fully realized. Ellipsis is the result of null lexicalization, not of deletion:

(14) Peter’s boss likes him a lot, and Mary’s boss does too.

The concept of iteration is similar to copy (in the copy theory of movement Nunes 1995), but there are some differences that make iteration a more interesting option. The copy operation creates an object that is identical to the source (there exists no difference between them).[9] A process of iteration can differentiate between "copies", since a determinate structure cannot be iterated, until it has been built before.

This is very important for an account of ellipsis, since there must be a mechanism that helps us differentiate between occurrences of the same VP sub-phrase marker, in order for ellipsis—null lexicalization in our terms—to apply. The phrase marker that has been built first is not redundant, thus, it will be overtly realized. The subsequent occurrences of the same phrase marker, however, are redundant and can be null lexicalized.[10]

For the first built VP, which involves iteration zero, null lexicalization does not apply. However, for the second built VP, null lexicalization does apply, as it will for any further iteration. The possibility of distinguishing between occurrences also

[9] In the examples that Nunes (1995) discusses (see (iii) below), it is possible to distinguish between copies and decide which one is going to be pronounced, because of the number of features that each copy has checked. For a sentence like (iii), both copies of the DP *which book* are identical (see (iv)a); however, there is still a Q feature in CP that needs to be checked. In ((iv)b) it can be observed how by making one more copy of the DP in the specifier of CP the Q feature is checked (both in the CP and in the wh-element). Thus, this last copy in [Spec, CP] is different from the others in terms of the number of features it has checked, and it is the copy that will be pronounced.

(iii) Which book did John file and Mary read?
(iv) a. [$_{CP}$ Q [$_{TP}$ John filed [**which book**]] and [$_{TP}$ Mary read [**which book**]]].
b. [$_{CP}$ **[which book]** [$_{TP}$ John filed **[which book]**] and [$_{TP}$ Mary read [**which book**]]].

In the case of verbal ellipsis, it is not clear how the copy theory of movement will help us decide which of the copies of the VP is going to be realized (see (v) below): there are two copies of the VP, but it cannot be determined which of them functions as the antecedent VP. The copies are identical in the number of features that hey have checked in this case, and there is no way of distinguishing them.

(v) John [**read that book**] and Mary did [**read that book**] too.

[10] The structural parallelism together with an emphatic feature in the numeration constraints null lexicalization of the iterated structure. In the presence of an emphasis feature in the numeration, the iterated structure will be lexicalized, in the absence of such a feature it will be null lexicalized.

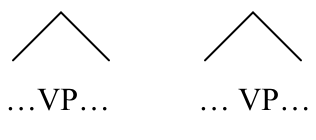
helps us linearize the parallel structures later on in the derivation, based on the antecedent-gap relation that holds between them.

3.2.3 Linearization of parallel structures

In Goodall's sense coordinated structures are parallel structures that arguably exist in separate, somehow parallel planes (see footnote 7). According to Chomsky (1995), the LCA can be reduced to an output condition on the shape of phrase markers at PF. Taking this into consideration, the fact that coordinate structures may be assembled in parallel is not an a priori problem, as long as they are linearized by the point they reach PF.

Even though there is an internal order among the elements that build each coordinate phrase marker (established by a MSO system), there is no obvious order between/among the coordinates themselves, since they are built in parallel. There is no command, hence no direct precedence relation between/among the parallel phrase markers. To converge, at least at PF, a structure must be linearizable. So some kind of order must be established, otherwise these structures will never converge. Observe (15):

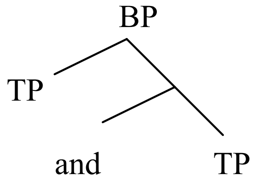
(15)



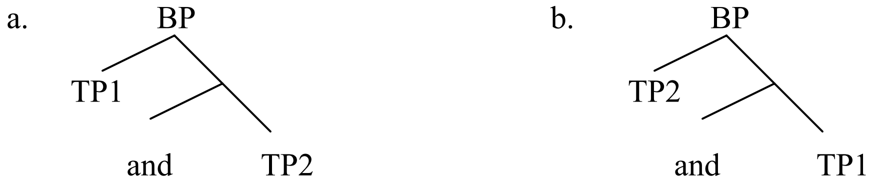
In (15) we have two parallel structures, to which no order has been assigned. In order to converge at PF some order must be assigned to the coordinates. I assume that coordinate structures project a Boolean Phrase (BP) (Munn 1987a) as in (16) below, where the coordinator heads its own phrasal projection.[11]

There is order among coordinates in a BP; in (16) the TP that is higher up in the tree c-commands and consequently precedes the lower TP. But, since coordinates are assembled in parallel, to only later on be linearized (because of PF necessities), the question is: how is order assigned to parallel coordinates? In principle, for two coordinate structures like the ones in (15) we could have as a result a Boolean Phrase as in (17) or as in (17):

(16)



(17)



[11] In the BP structure I am assuming—headed by the coordinator—the coordinates occupy the specifier and the complement position, essentially as in Kayne (1994). I am not assuming that the BP is formed by the coordinator and the second coordinate—which are in turn adjoined to the first coordinate—as Munn proposes in 1993.

In (17) and (17) we have the opposite command relations. In the first case it is TP1 that c-commands and precedes TP2. However, in the second case it is TP2 which c-commands and precedes TP1. I suggest that the order in which these two phrase markers are linearized comes as a result of null lexicalization affecting one of the two, and from the relationship of antecedence-gap that holds between both.[12] The one which null lexicalization does not affect comes first. This one functions as the antecedent to, in some sense, interpret the second. The final object is the one in (18):[13]

[12] Goodall (1987) proposes an ordering relation for coordinate structures, which is based on the existing order among elements on each of the coordinate sub-parts. To be precise, the order of a sentence like (vi) is derived from the order that every single element has in each of the sub-parts in (vii):

(vi) John and Mary eat doughnuts (= ((vii)a) U ((vii)b))
(vii) a. John ate doughnuts.
b. Mary ate doughnuts.

Both in (vii)a and (vii)b, *ate* precedes *doughnuts*. In (vii)a *John* precedes *ate* and *doughnuts*, and in (vii)b *Mary* precedes both elements as well. Consequently, the order assigned to the coordination of (vii)a and (vii)b is that one in (vi), since it respects all the previous precedence relations in each of the subparts. We do not assume this type of ordering relation, because it is not clear how it will assign order to a sentence like (viii), when its subparts are those in (ix):

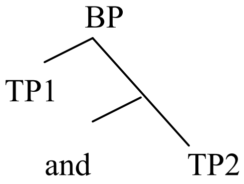
(viii) John ate cookies, and Mary did too.
(ix) a. John ate cookies.
b. Mary ate cookies.

[13] Sentence (x) seems to argue against this antecedent-gap ordering relation, since in this case the antecedent comes first and the gap follows:

(x) Peter didn't, but Mary went to the party.

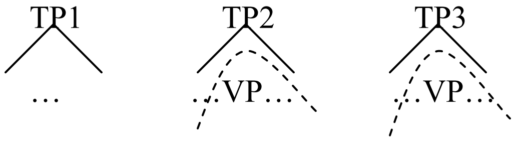
I would like to argue that this sentence has been derived in the same way as the previous sentences. It has been also linearized following the antecedent-gap relation, but then the elided clause has moved higher up in the tree to a Focus position—sentences like this one are not neutral, the material in the elision site is focused.

(18)



However, this is not the end of the problem for linearization. In sentences where more than two phrase markers have been coordinated, an antecedent-gap relation is clearly not enough, because there is one antecedent clause, but two or more clauses are affected by null lexicalization, as in (19):

(19)



One possibility, which I assume here, is that the system somehow keeps track of the order in which the different phrase markers have been built. They are kept in a stack and linerized in a First-in Last-out order (FILO).

To summarize, lexical insertion is a late process and coordinated structures that happen to have undergone VPE can be reanalyzed in terms of null lexicalization which affects an iterated structure, because it is redundant. Coordination creates parallel structures, which only later on, are ordered with respect to one another, thus meeting linearization requirements.

3.2.4 Fiengo and May (1994) Revisited

3.2.4.1 Sloppy and Strict Readings: before and after Lexical Insertion

With a system in which lexicalization is a delayed process in the derivation, we can account for the readings that F&M (1994) described (see Chapter 2, section 2.3.2), without the need to postulate different index types (α- or β- occurrences). To remind the reader which the readings are, I exemplify them below:

(20) Max saw his mother, and Oscar did too.

(21) a. Max saw Max's mother, and Oscar saw Oscar's mother. *(sloppy)*

b. Max saw Max's mother, and Oscar saw Max's mother. *(strict)*

In the system proposed here, null lexicalization applies to structurally parallel VPs (see section 3.2.2 of this chapter), so in order to interpret the elided—null lexicalized—VP, the overt VP functions as antecedent—material must be copied from the antecedent VP into the elided VP at some point in the derivation.[14]

We can account for strict and sloppy readings depending on which moment in the derivation the elided VP is interpreted, that is to say, prior to or after lexical insertion. If this interpretation process takes place before lexical items are inserted, the syntactic object is constituted by categories only: DP, V, NP, etc. I will assume that the pronoun DPs are then mere variables, which have not yet been assigned a

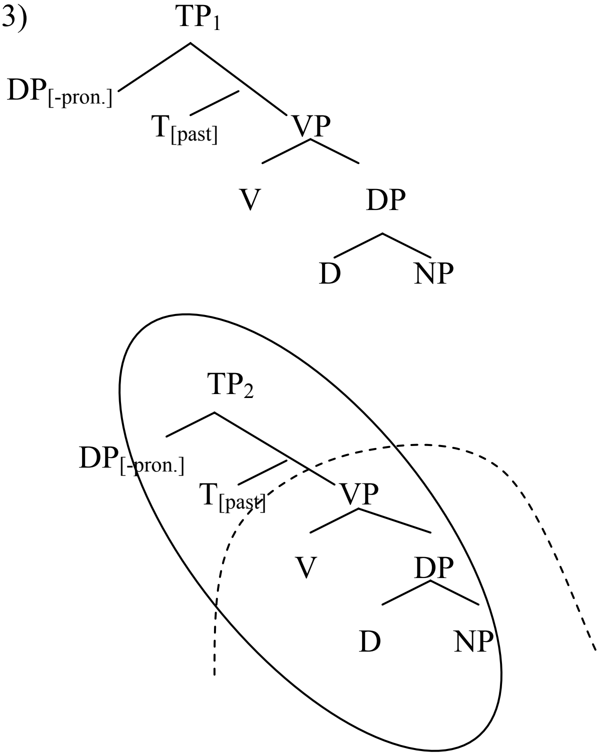
[14] The iteration process (which creates a VP skeleton) should not be confused with the copying/interpretation process that I will discuss now. After the phrasal skeleton is built, substantive representations are added, by copy at LF.

value. However, if the interpretation is done after lexical insertion, then it can clearly also involve lexical items form the antecedent VP like: *see*, *his*, etc. The pronoun DPs are then constants, having been assigned a value through lexical specification. Consider how this system works with example (22):

(22) Max saw his mother and Oscar did too.

The numeration starts with: {$D_{[\text{-pronominal}]1}$, $D_{[\text{-pronominal}]1}$, $T_{[\text{past}]2}$, V_1, $D_{[\text{+pronominal}]1}$, N_1, B_1}. Once both phrase markers have been built, we have something like (23), to which null lexicalization will apply for the second conjunct VP. At this point—when the phrasal skeleton for both parallel phrase markers has been built—the option of copying-for-interpretation from the first into the second conjunct exists, but there is no lexical material to be copied yet. If the interpretation is done at this stage in the derivation, then the sloppy reading emerges.

(23)



The sloppy reading for (22) would emerge if the second phrase marker (TP$_2$) in (23) is sent to LF for interpretation before lexical insertion takes place—when the coordinates are still parallel structures. Of course, LF will not be able to assign a complete interpretation to this structure, since lexical items have not been inserted yet. If the pronouns in the elided VP are interpreted in parallel to the ones in the antecedent VP, then the sloppy reading emerges. If they are not (if the SD in F&M's terms is different) then this kind of reading is precluded (I will come back to this in section 3.2.4.4).

LF partially interprets the structure that has been sent, it interprets it as much as possible.[15] The syntactic structure so far is built of categories—open variables which are assigned a value through lexical specification—so one way of interpreting this structure is by assigning a value to the variables wherever it is possible.

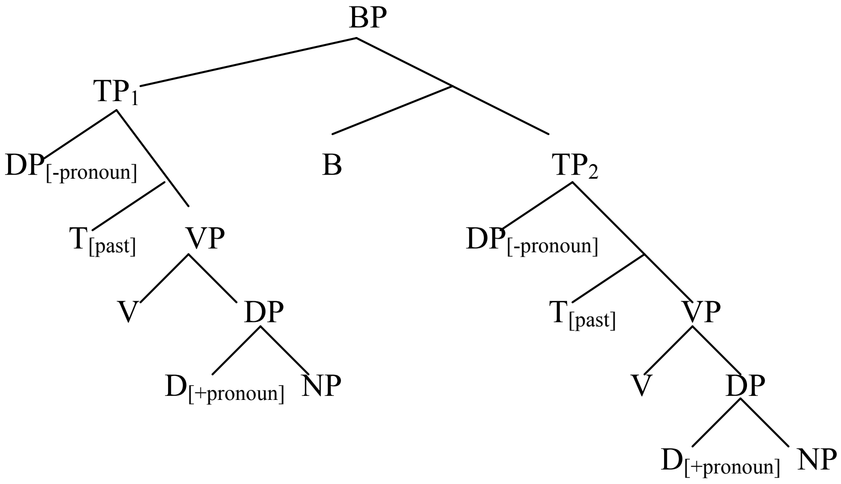
The DP pronouns are variables, but by binding a DP pronoun to some other element it is assigned a value—the one that the binder will obtain through lexical specification.[16] The DPs are bound by a local potential antecedent—one that is inside the same phrase marker. Thus, in (23) the D pronoun in the elided VP is bound by the DP subject—it is assigned the value of the DP subject, receiving a bound variable interpretation.

If instead of interpreting at the stage where lexical items have not been inserted (in (23)), the two parallel structures are linearized (as in (24)), lexical items inserted (as in (25)), the whole BP is sent to PF and LF, and the elided VP is interpreted at this later stage (at LF) (as in (26)), then the strict reading comes out. In order to interpret the elision site, lexical material from the antecedent is borrowed and copied into the ellipsis. When we copy from the first into the second VP, we copy: *saw*, *his*, and *mother*; we copy constants and we get the strict reading:

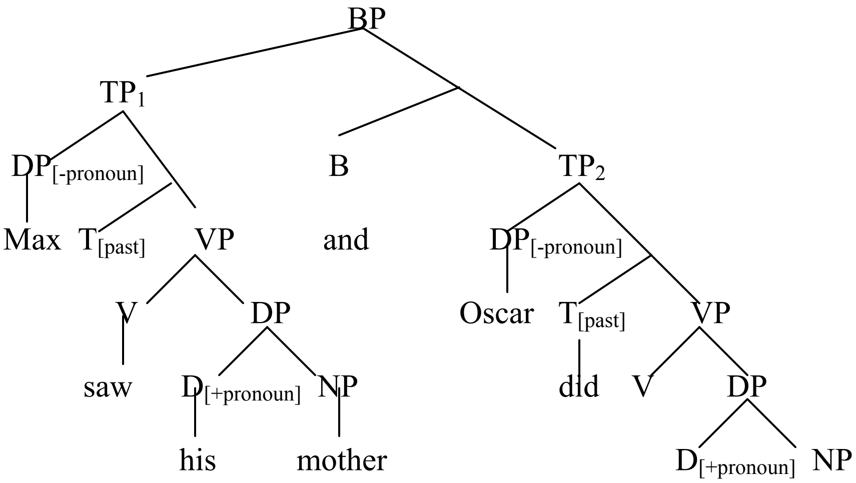
[15] What LF does at this point is a partial interpretation of the structure, much along the lines of the partial LF scope interpretation that Fox proposes (Fox 1999).

[16] As a result of the dependency established for the pronoun, it is interpreted as a β-type index (in F&M 1994). The pronoun gets its value (reference) from the element it is connected to, not independently.

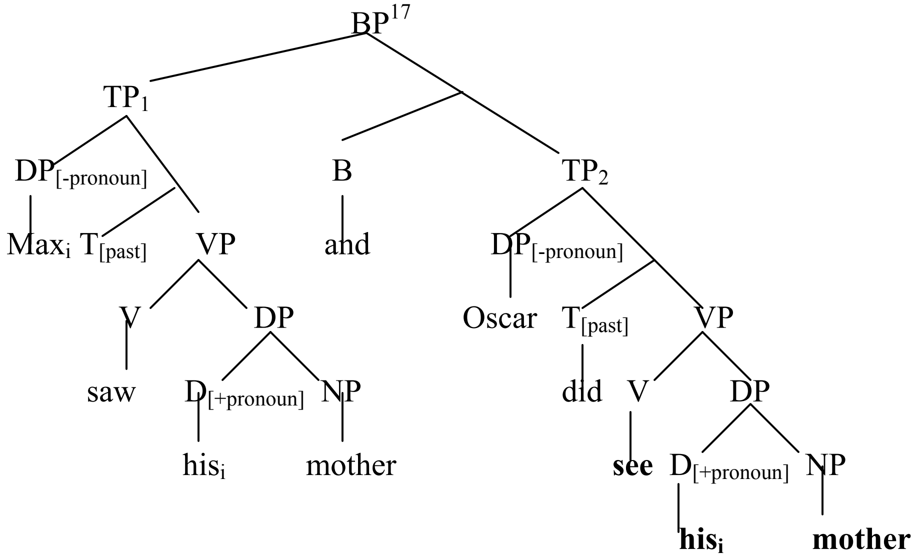
(24)


BP
TP1
B
TP2
DP[-pronoun]
T[past]
VP
DP[-pronoun]
V
DP
T[past]
VP
D[+pronoun]
NP
V
DP
D[+pronoun]
NP

(25)


BP
TP1
B
TP2
DP[-pronoun]
Max
T[past]
VP
and
DP[-pronoun]
V
DP
Oscar
T[past]
VP
saw
D[+pronoun]
NP
did
V
DP
his
mother
D[+pronoun]
NP

(26)



3.2.4.2 The Many-Pronoun-Puzzle Revisited

In sentences like (27) or (29), where more than one pronoun has been elided, ellipsis has an eliminative effect on the combinatorial anaphoric possibilities. The number of readings that emerge in such sentences is smaller than in their non-elided counterparts—for these last all the readings listed in (28) and (30) are acceptable:

(27) Max said he saw his mother, and Oscar did too.

(28) a. Max said Max saw Max’s mother and Oscar said Oscar saw Oscar’s mother. (*across-the-board sloppy*)

[17] In order to interpret the elided VP at LF, we copy the lexical items from the antecedent into the elision site. One question that can be raised is how the verb can show different forms in the antecedent and in the gap. The verb *to sleep* appears as *saw* in the antecedent, but its corresponding form in the elision is *see*, since the past is already marked by the auxiliary *did*. For a discussion on how this is possible I refer the reader to section 3.3.2.

b. Max said Max saw Max’s mother and Oscar said Max saw Max’s mother. (*across-the-board strict*)

c. Max said Max saw Max’s mother and Oscar said Oscar saw Max’s mother. (*sloppy-strict*)

d. *Max said Max saw Max’s mother and Oscar said Max saw Oscar’s mother. (*strict-sloppy*)

(29) Max said he thinks he saw his mother and Oscar did too.

(30) a. Max said Max thinks Max saw Max's mother, and Oscar said Oscar thinks Oscar saw Oscar's mother. (*across-the-board sloppy*)

b. Max said Max thinks Max saw Max's mother, and Oscar said Max thinks Max saw Max's mother (*across-the-board strict*)

c. Max said Max thinks Max saw Max's mother, and Oscar said Oscar thinks Oscar saw Max's mother. (*sloppy-sloppy-strict*)

d. Max said Max thinks Max saw Max's mother, and Oscar said Oscar thinks Max saw Max's mother. (*sloppy-strict-strict*)

e. * Max said Max thinks Max saw Max's mother, and Oscar said Oscar thinks Max saw Oscar's mother. (*sloppy-strict-sloppy*)

e. * Max said Max thinks Max saw Max's mother, and Oscar said Max thinks Max saw Oscar's mother. (*strict-strict-sloppy*)

f. * Max said Max thinks Max saw Max's mother, and Oscar said Max thinks Oscar saw Oscar's mother. (*strict-sloppy-sloppy*)

g. * Max said Max thinks Max saw Max's mother, and Oscar said Max thinks Oscar saw Max's mother. (*strict-sloppy-strict*)

F&M accounted for the available readings in terms of (i) the reconstruction of different index types and (ii) the identity conditions that they must respect.

I want to argue that the reason we can have some combinations of readings, but not others is that the interpretation process can involve one or more pronouns at each stage in the derivation. But the way in which this structure interpretation takes place cannot be by randomly interpreting chuncks of structure at LF. It is a cyclic operation. By cyclic here I do not refer to the familiar concept of cyclicity, established bottom-up of the tree when building a syntactic structure and that is directly related to constituency. Observe (31):

(31) a. [$_{IP}$ The man [$_{VP}$ said [$_{IP}$ he had [$_{VP}$ the time [$_{PP}$ of his life]]]]].

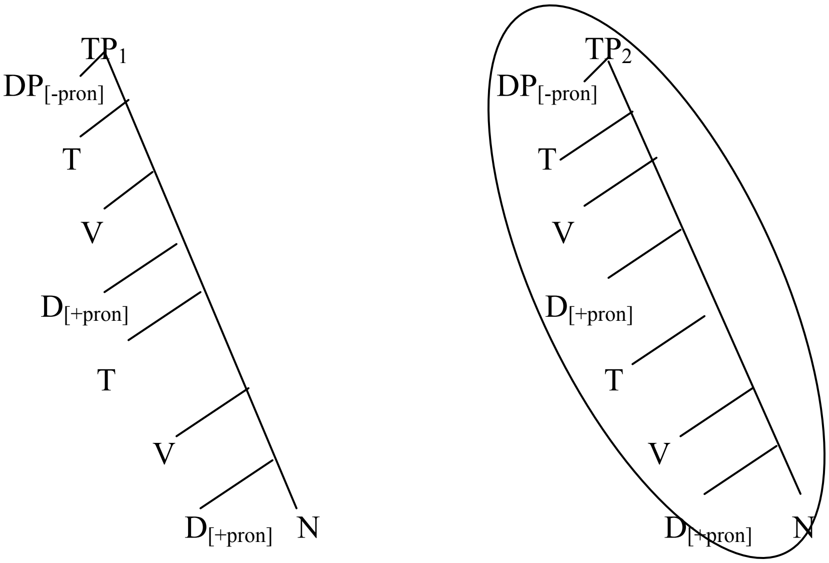
b. [[The man said he] had the time of his life].

Both (31) and (31) are representations of different possible cyclic relations. In (31) the cycles are established bottom-up in the tree (right-to-left)—representing some of the constituents in that structure. In (31), however, the cycles are not established on a constituent basis, at least in traditional terms (a system like that proposed by Philips, 1996; Drury, 1998; or Guimaraes 1999 which builds the syntactic tree in a top-down fashion, establishes cycles and constituents in this way), they are established top-down in the tree (left-to-right). This is the kind of cyclic relation I am talking about here.

I want to argue that—independently of how the syntactic structure has been built, top-down or bottom-up—antecedence relations are established in a top-down fashion, following the precedence-command order, like linearization does. This move may be justified in information-theoretic ways.

Consider example (27) again. In order to get the across-the-board SL reading, prior to lexical insertion, the whole structure for the second phrase marker is sent to LF and interpreted. Both pronoun DPs in the second VP will be bound by the only local DP available, the DP subject in that phrase marker as in (32):

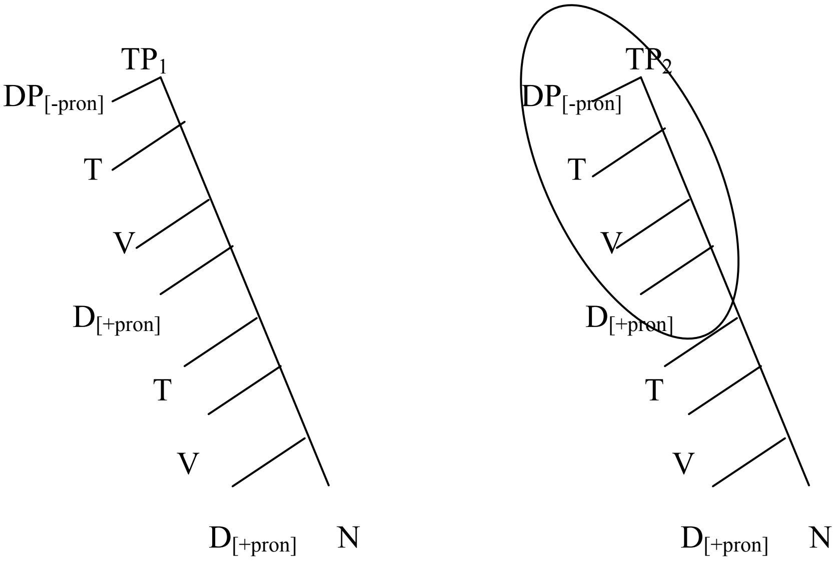
(32)



In order to get the across-the-board strict reading we just have to copy all the material (including pronouns and pronouns reference) from the first into the second conjunct after lexical insertion.

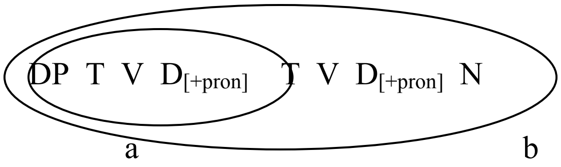
For the (sloppy-strict) reading only part of the syntactic structure for the second phrase marker is interpreted at LF—that part that contains from the subject DP to the first pronoun (see (33) below). The second pronoun is not interpreted until later on, when both the first and the second phrase marker have been lexicalized and sent to LF—the copy is done at this stage.

(33)



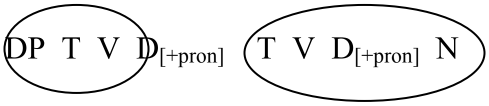
Observe (34). If the structure that is sent to LF and interpreted is that one inside circle ‘a’, then the (sloppy-strict) reading emerges. If the structure interpreted at LF is that one inside ‘b’ then the across-the-board reading does:

(34)



However, we cannot get the (strict-sloppy) reading, because this would be a counter-cyclic operation. It is important to remember that we have claimed antecedence relations are established top-down of the tree. To get this last reading part of the structure will have to be ignored—some part of the structure in the second VP (at least that part involving the first elided pronoun in the VP) will not be interpreted at LF, violating cyclicity:

(35)



If antecedence relations are established top-down, the first pronoun in the VP should be interpreted first, followed by the second pronoun. But, in order to get the (strict-sloppy) reading this order has to be broken. The system can operate with part of the structure, but I am assuming the part chosen has to be a linear continuum. It cannot affect discontinuous chunks of structure from the top and from the bottom of the tree.

The same kind of reasoning applies to (29)—the readings in (30) that are not acceptable violate cyclicity so they are not possible.

3.2.4.3 The Many-Clause-Puzzle Revisited

So far we have discussed examples where there are two coordinates; one antecedent and one gap. We consider in this section examples that involve more than two coordinates, i.e. more than one gap. Sentences such as (36) have been claimed to be interpreted only as strict or sloppy across-the-board. No mixed readings are possible, according to F&M (see Chapter 2, section 2.3.1). Thus, once more, the effect of ellipsis on anaphoric possibilities is eliminative—in the non-elided counterparts more readings are available:

(36) Max saw his mother, Oscar did too, but Sam didn't.

(37) a. Max saw Max's mother, Oscar saw Oscar's mother, but Sam didn´t see Sam's mother. *(across-the-board sloppy)*

b. Max saw Max's mother, Oscar saw Max's mother, but Sam didn't see Max's mother. *(across-the-board strict)*

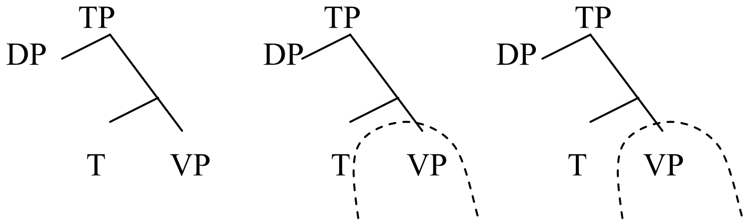
c. *Max saw Max's mother, Oscar saw Max's mother, but Sam didn't see Sam's mother. *(strict-strict-sloppy)*

d. *Max saw Max's mother, Oscar saw Oscar's mother, but Sam didn't see Max's mother. *(sloppy-sloppy-strict)*

F&M (1994) argue that the reason for the non-existence of mixed readings is that they would involve the reconstruction of an α-type occurrence as a β-type or vice versa. In the system we are pursuing here, this is a natural consequence: again coordinate structures are parallel structures, to which syntactic operations apply in parallel. Thus, if the second VP in (36) is interpreted as strict or sloppy—that is, before or after lexical insertion respectively—the same must apply to the third conjunct.

Let us consider the derivation of sentence (36) with some detail. From the numeration: $\{D_{[-pronominal]1}, D_{[-pronominal]1}, D_{[-pronominal]1}, T_{[past]3}, v_1, D_{[+pronominal]1}, N_1, B_2\}$, three parallel structures are built. Two of them are going to be null lexicalized. The resulting object is the one in (38):

(38)



The interpretation for the second and the third conjunct can be made before lexical insertion, or after lexical insertion—giving the two different interpretations: across-the-board sloppy or strict, respectively. But, the three coordinates have to be interpreted in parallel, that is, we cannot have one conjunct interpreted as sloppy and another as strict or vice versa. Or in other words, if one coordinate is interpreted at a certain stage the other coordinates must be interpreted too.

Thus, the across-the-board sloppy and strict readings are explained as follows: if we send one coordinate to LF before lexical insertion, the same operation must apply to the rest of the parallel elided VPs—resulting in the sloppy reading.

Otherwise, all are interpreted after lexical insertion by copying substantive material from the antecedent to the elided VPs—resulting in the strict reading.

3.2.4.4 Sloppy Readings and Structural Parallelism

I argued above that in a framework like the one I have presented here, the phrase-marker that contains the elided VP has to be sent to interpretation before lexical insertion takes place, in order to get the sloppy interpretation. But I also mentioned that the interpretation of the pronouns in the elided VP must be parallel to that of the pronouns in the antecedent VP—it must be structurally parallel (the dependencies must involve the same SD in F&M's terms). If both interpretations are not parallel, then the sloppy reading is precluded. Consider sentence (39) and its readings in (40), an example of the lack of structural parallelism blocking a sloppy reading:

(39) Max saw his mother, and Oscar said that Harry did too.

(40) a. Max saw Max´s mother, and Oscar said Harry saw Max´s mother.

b. Max saw his mother, and Oscar said Harry saw his mother.

c. *Max saw his mother, and Oscar said Harry saw his mother.

Sentence (39) can be given both a strict and a sloppy interpretation, represented in (40) and (40) respectively. However, the sloppy reading in (40) where the pronoun *his* in the elided VP is interpreted as referring to *Oscar* and not to *Harry* is not a possibility. The sloppy pronoun in the elision cannot refer to Oscar; the reason is that in such a reading the pronoun indexical dependencies in the first and second VP are not structurally parallel.

Thus, it is necessary to impose a further parallelism constraint on the sloppy interpretation mechanism. The system would need to check if the dependencies established in both the antecedent and the elided VP are parallel or not, in order to accept or reject a sloppy interpretation under ellipsis. The system interprets the elided VP, and later on checks if the dependencies established for this are structurally parallel to the dependencies in the antecedent conjunct.[18]

3.3 Identity of Syntactic Structures: Identity of Categories

The issue of the syntactic identity condition between antecedent and gap that must be respected in order for ellipsis to apply has been raised in different occasions and guises. In the previous chapter, we discussed some of the examples that, in principle, seem to argue against such a restriction: (i) Lasnik (1995b) examines examples where there is a difference in verbal morphology between antecedent and gap (see (41)); and (ii) F&M's (1994) vehicle change theory which analyzes examples where the difference is between pronouns (see (42)), or pronouns and reflexives (see (43)). In

[18] It looks like a "double identity" is needed for VPE to apply. On the one hand, parallel syntactic structures between antecedent and gap must be found before lexical insertion takes place. On the other hand, at LF the dependencies that are established in the antecedent and in the elision must be either identical (the case of the strict reading) or structurally parallel (the case of the sloppy reading). It could be asked why that should be the case, why this double identity is a requirement. I would like to suggest that it could be the case that these two different identities are needed for PF and LF purposes, but I will not pursue the matter here.

both cases, an analysis that accounts for the differences and which maintains the identity condition is advanced:

(41) John slept and Mary will ~~sleep~~ too.

(42) John saw his mother and Mary did ~~see her mother~~ too.

(43) John admires himself and Mary does ~~admires herself~~ too.

In the previous section, I introduced a proposal that accounts for strict and sloppy readings under ellipsis which is based on the notions of late lexical insertion and of identity of syntactic categories. The question I want to raise now is whether that account is extendable to the examples in (41)-(43). In other words, can we account for what seems to be partial syntactic identity with the proposal sketched, and thus, offer a unified treatment of all those examples that seem to threaten the condition of identity of syntactic structures for ellipsis? I want to argue that this is possible, and also to show that syntactic identity is always respected. Rather than arguing against identity, what the examples in Chapter 2 demonstrate, is that a more abstract notion of identity is needed. I propose that this new notion corresponds to that of syntactic categories, which can be satisfied before lexical insertion takes place.

3.3.1 Identity on Ellipsis, and Syntactic Categories and Features

Before getting into the examples of partial identity in detail, the question of whether it is just syntactic categories alone, or syntactic categories and syntactic features together that are relevant for identity should be raised. And also, do features enter the derivation underspecified and take their value in each conjunct, or do these features already have a value in the numeration? To answer the first question (whether syntactic feature values have to be identical or can differ), consider the examples below with the reading in parentheses:

(44) Max saw his friend, and Peter did too. (= Peter$_i$ saw his$_i$ friend)

(45) Max saw his friend, and Susan did too. (= Susan saw her friend)

(46) Max shaved himself, and Susan did too. (= Susan shaved him)

(47) Max reads lots of novels, and I do too. (= I read lots of novels)

The differences between syntactic features do not seem to be important in any of the sentences above. The first sentence, example (44), is grammatical; in this case there are no differences in the features from the antecedent to the gap. That is not the case with the rest of the examples. In (45), there is a gender difference in the pronouns: masculine in the antecedent and feminine in the gap. In (46), there is not only a gender difference, but also a difference in the nominal features: the reflexive pronoun in the antecedent is realized as a pronoun in the gap. In (47), a difference can be observed in the verb person feature: 3rd person singular for the antecedent, and the bare unmarked form for the gap. Let us conclude for the time being that syntactic features are not relevant for identity, though we will see when verbal morphology in Spanish is discussed that this generalization has to be slightly modified.

In order to answer the question of whether these features are underspecified or they enter the derivation with a value, we need a more detailed study than the one that is offered here. We are just going to concentrate on a few examples. Many more cases should be studied, and not only ellipsis cases, but also other types of constructions, so that a generalization can be made with respect to syntactic features and to how they enter the derivation (i.e. with or without a value). Besides, it could be the case that different languages, or even different speakers of the same language, instantiate features in slightly different ways. I believe this to be an interesting issue to be studied, but it is not the purpose of the present work to do so.

I opt for having feature values instantiated, since there is some evidence that seems to argue in that direction. Kitagawa (1991) (see section 2.2.2) observes that in sentences with reflexives, if there are no gender conflicts in the elision between the subject and the reflexive (see (48)), then the sloppy reading is preferred (i.e. it is the reading that all speakers get). However, when there are differences in gender (see (49)), the strict reading is preferred over the sloppy one, since gender conflicts are avoided. One way in which this effect on change in preference can be accounted for is if the features have a value assigned.

(48) John considers himself to be intelligent, and Bill does too.

(49) John considers himself to be intelligent, and Mary does too.

3.3.2 Verbal Morphology

Lasnik (1994) discusses examples of VPE where differences in verbal morphology between antecedent and elision can be observed (see (50), (51) and (52) below). He argues that ellipsis takes place under identity by the application of a rule of deletion at PF, before Affix Hopping (see Chapter 2, section 2.1 for the whole discussion). Thus, he accounts for the differences in verbal morphology in the sentences below:

(50) John slept, and Mary will ~~sleep~~ too.

(51) John sleeps and Mary will ~~sleep~~ too.

(52) John has slept and Mary will ~~sleep~~ too.

We can account for these differences with our proposal too. In all three examples, the syntactic identity condition is respected; even though superficially there seem to be morphological differences in the verbal forms (e.g. *slept* versus *sleep* in (50)). The stage of the derivation at which the identity condition needs to be met is before lexical insertion, that is, when just syntactic categories constitute the phrasal skeleton. So even though those categories would show differences if they had been lexicalized, at the point where identity of structure is checked only a category V is found in the antecedent and in the gap. Let us discuss one example in some more detail.

Sentence (50) is built out of the Numeration in (53) below, which is formed by mere syntactic categories and syntactic features. The derivation starts out with those categories, and after several merge and move operations the sentence has the structure in (54), where the syntactic skeleton for both coordinates is already assembled. At this stage in the derivation, identity of syntactic structure is met and Null Lexicalization applies to the elided VP. After lexical items are inserted, the sentence will look like that in (55), where the elided verb phrase is formed by a category V, but no lexical item.

(53) $\{T_{[+past]1}, T_{[+future]1}, D_{[-pronoun]2}, V_1, B_1\}$

(54) [$_{TP}$ DP$_k$ T$_{[+past]}$ **[$_{VP}$ t$_k$ V]**] and [$_{TP}$ DP$_j$ T$_{[+future]}$ **[$_{VP}$ t$_j$ V]**].

(55) [$_{TP}$ John$_k$ T$_{[+past]}$ [$_{VP}$ t$_k$ slept]] and [$_{TP}$ Mary$_j$ will [$_{VP}$ t$_j$ V]] too.

When the elided VP is interpreted at LF (after the PF and LF split), the lexical material from the antecedent will be borrowed, and copied into the gap. The lexical material in the antecedent is used to interpret the gap because (i) no lexical items are inserted in the gap, and (ii) the antecedent and the elided VP are occurrences of the same tokens in the numeration. The fact that both verbal categories are occurrences of the same token guarantees that they contain the same lexical item, in other words, that both will be lexicalize as *sleep*, not as *sleep* and *arrive* for example.

One question that could be raised now is how the verb *sleep* is realized as *slept* in the antecedent and as *sleep* in the gap. Lasnik (1995b) explains this difference as a result of Affix Hopping—an operation that takes place at the level of PF (see discussion on Chapter 2, section 2.1). However, we can account for it in a different way: both verbs are occurrences of one token in the Numeration; therefore, they have to be filled with the same lexical item, the verb *sleep*. In the antecedent it is realized as *slept*, since the syntactic feature for Tense is [+Past], while in the gap it is realized as *sleep*, because the Tense feature in this case is [+Future]. Thus, the lexical items from the antecedent are borrowed and copied into the gap, but respecting the new syntactic context in which they appear. The interpretation process allows for this type of flexibility, while respecting lexical identity at the same time.

Having discussed the case of main verbs and ellipsis, let us now consider some examples with auxiliaries and see how the proposal presented here can account for them. Lasnik observes that there can not be any morphological difference between auxiliaries in the antecedent and elision. He discusses the examples in (56), and (57). The examples in ‘c’ are added:

(56) a. ?John should have left, but Mary shouldn’t ~~leave~~.

b. *John should have left, but Mary shouldn’t ~~have left~~.

c. John should have left, but Mary shouldn’t have ~~left~~.

(57) a. John has left, but Mary shouldn’t ~~leave~~.

b. *John has left, but Mary shouldn’t ~~have left~~.

c. John has left, but Mary shouldn’t have ~~left~~.

We can also account for these cases if we take into account a difference between main verbs and auxiliaries in English: auxiliaries occupy the node T (whether they are base generated there or raise to that position is not crucial for the argument here), while main verbs remain in V. Taking this into account, it seems we can draw the following conclusion from the examples with auxiliary *have*: everything that is under VP can be deleted (even if it is different (e.g. *leave* versus *left*)), since what is relevant is identity of categories), what is under TP cannot be deleted. This is the reason why the "a" and "c" cases are acceptable, but not the "b" cases, since in this last case some material that resides in T (*have*) has been affected by elision.

In order to close our discussion on ellipsis and verbal morphology I consider now some examples of verbal ellipsis in Spanish, which raise an interesting contrast. In Spanish, together with the verbal phrase, the whole Tense projection is elided.[19] Here, as in English, agreement differences in the verbal form, such as number and person, can be found (see (58) and (59)). But, contrary to English, the whole TP is deleted and Tense differences are not allowed, see (60):

(58) Nosotros dormimos poco y vosotros ~~dormís poco~~ también.
We sleep-1st sing little and you ~~sleep-2nd pl little~~ too.
We sleep a few hours, and you do too.

[19] See footnote 8 for a brief discussion on why it could be that it is TP instead of VP that is affected by elision in Spanish.

(59) Has viajado a muchos países y ellos ~~han viajado a muchos países~~ también.

Have-2nd sing traveled to lots of countries and they ~~have-3rd pl traveled to~~ ~~lots of countries~~ too.

You have traveled to lots of countries, and they have too.

(60) *María ha leído mucho y Elena en el futuro ~~habrá leído~~ ~~mucho~~ también.

María have-3rd sing read a lot, and Elena in the future will ~~have-3rd sing read~~ ~~a lot~~ too.

Mary has read a lot, and Elena in the future will have too.

Our proposal of identity of syntactic categories can also account for the verbal differences (e.g. *dormimos* versus *dormís*) found in the examples above. In these examples, as well as in the English ones, when identity of syntactic structure is checked for, the sentence structure is built out of syntactic categories; no lexical items have been inserted yet, so identity is met.

However, it turns out that Tense differences are not allowed in Spanish. One question that could be raised is why it should be the case that person and number agreement feature differences are irrelevant, while Tense differences matter for ellipsis. A reason for this contrast can be found in the nature of the features themselves: person and number are agreement, that is, relational features; while strictly Tense is not. Arguably only relational features can change/adjust values in ellipsis context (i.e. to agree), without violating the constraint on identity.

If that is the correct conclusion, we need to slightly modify our previous generalization about identity. Both syntactic categories and syntactic features like Tense are relevant for identity under ellipsis.

3.3.3 Agreement and Nominal Features

We turn now to those examples exhibiting the following differences between antecedent and elision: masculine pronoun/feminine pronoun, feminine pronoun/masculine pronoun, pronoun/reflexive, reflexive/pronoun, name/pronoun, name/reflexive. We have reviewed two analyses that deal with these differences: F&M (1994) and Kitagawa (1991).

3.3.3.1 Some Representative Examples

F&M account for all of the examples in terms of an operation called Vehicle Change (see Chapter 2, section 2.3.4), which they argue allows for free adjustment of feature values. Kitagawa also studies some of these cases (see Chapter 2, section 2.2.2). Although his analysis is restricted to pronoun/pronoun and reflexive/pronoun variations, he obtains a nice generalization with respect to the readings that are allowed in different dialects under different circumstances—i.e. in ellipsis sentences that involve gender conflict and others that do not. He accounts for ellipsis in terms of the features that are copied from the antecedent into the gap at LF—features can be copied optionally. Consider example (61), (63), and (65). The favored reading for (61) is the sloppy reading in (62). Instead, for sentences (63) and (65), the favored reading is the strict one in (64) and (66), respectively. For the last two sentences, in the case of the strict reading the feature [+anaphor] is not copied, as opposed to the sloppy reading where the non-copied feature is [+masculine] or [+feminine].

(61) John considers himself to be intelligent, and Bill does too.

(62) a. John$_i$ considers [+anaphor, +pronoun, +masculine]$_i$ ... , and
Bill$_k$ considers [+anaphor, +pronoun, +masculine]$_k$ …

b. John$_i$ considers [**+anaphor**, +pronoun, +masculine]$_i$… , and
Bill$_k$ considers [**ø**, +pronoun, +masculine]$_i$…

(63) John considers himself to be intelligent, and Mary does too.

(64) a. John$_i$ considers [+anaphor, +pronoun, **+masculine**]$_i$ ... , and
Mary$_k$ considers [+anaphor, +pronoun, **ø**]$_k$…

b. John$_i$ considers [**+anaphor**, +pronoun, +masculine]$_i$… , and
Mary$_k$ considers [**ø**, +pronoun, +masculine]$_i$…

(65) Mary considers herself to be intelligent, and Bill does too.

(66) a. Mary$_i$ considers [+anaphor, +pronoun, **+femenine**]$_i$ ... , and
John$_k$ considers [+anaphor, +pronoun, **ø**]$_k$…

b. Mary$_i$ considers **[+anaphor**, +pronoun, +femenine]$_i$… , and
John$_k$ considers [**ø**, +pronoun, +femenine]$_i$…

Abstracting away from specific dialectal variations, we can summarize Kitagawa's observations by saying that speakers in general have a preference not to copy the feature [+/- anaphor] over [+masculine/feminine]. As it was mentioned when discussing Kitagawa's proposal, the reason why this contrast in the preference of one reading over another is found should be accounted for. In particular, it ought to be clear why the strict reading is preferred in examples (63) and (65); in other words, why optionality of copying feature [+ anaphor] is preferred to optionality of feature [+masculine] or [+feminine].

We can account for that contrast if we assume the classification of nominal features that F&M propose. According to them, there is only one feature [+/-pronoun], but no feature [+/-anaphor]: both reflexives and pronouns belong to the same nominal typology and are characterized as [+pronoun]. Reflexives are composite elements, built of two parts: an argumental part, and the grammatical formative *self* that triggers Principle A of the Binding Theory. Taking this classification into account, it can be explained why there is a preference for the strict reading in sentences (63) and (65) above—no feature value needs to be modified, while for the sloppy reading the value of the gender feature (in bold) needs to be modified. Therefore, a preference for a strict interpretation of the elision site is observed:

(67) John considers himself to be intelligent, and Bill does too.

(68) a. John$_i$ considers himself [+pronoun, +masculine]$_i$ ... , and
Bill$_k$ considers himself [+pronoun +masculine]$_k$…

b. John$_i$ considers himself [+pronoun, +masculine]$_i$… , and
Bill$_k$ considers him [+pronoun, +masculine]$_i$…

(69) John considers himself to be intelligent, and Mary does too.

a. John$_i$ considers himself [+pronoun, **+masculine**]$_i$ ... , and
Mary$_k$ considers herself [+pronoun **-masculine**]$_k$…

b. John$_i$ considers himself [+pronoun, +masculine]$_i$… , and
Mary$_k$ considers him [+pronoun, +masculine]$_i$…

(70) Mary considers herself to be intelligent, and John does too.

(71) a. Mary$_i$ considers herself [+pronoun, **-masculine**]$_i$ ... , and
John$_k$ considers himself [+pronoun **+masculine**]$_k$…

b. Mary$_i$ considers herself [+pronoun, +masculine]$_i$… , and
John$_k$ considers her [+pronoun, +masculine]$_i$…

F&M account not only for the examples (61), (63), and (65) with their theory of Vehicle Change, which allows for adjustment of feature values, but also for examples like the ones below:

(72) I turned in my assignment, but most of the students didn't ~~turn in their assignments~~.

(73) Mary loves John$_i$, and he$_i$ thinks that Sally does ~~love him$_i$~~ too

(74) I shaved John$_i$, because he$_i$ wouldn't ~~shave himself~~$_i$.

3.3.3.2 How the Current Proposal Accounts for the Previous Examples

We can also account for the previous examples with our theory of identical syntactic structures. In all the examples above, before lexical insertion takes place the syntactic structure is the same. There are differences in some cases in agreement and nominal features, but these differences do not affect the condition on identity.

Consider example (75) first, where no feature value needs to be changed neither for the sloppy nor for the strict reading. For the sloppy reading in (76), both reflexives in the antecedent and in the elision site are masculine, so there is no need for feature change. Interpreting the sentence before lexical insertion takes place gives the sloppy reading as a result, where each reflexive is bound in its own clause. For the strict reading in (76) (after lexical insertion), the reflexive in the antecedent needs to be realized as a pronoun in the elision site—it cannot be a reflexive, since its antecedent *John* does not bind it in this position—but this does not involve any change in feature value:

(75) John likes himself, and Peter does too.

(76) a. [$_{TP}$ [$_{DP}$ D[-pron, +masc]]$_i$ T$_{[present]}$ [$_{VP}$ t V [$_{DP}$ D[+pron, +masc]]]]$_i$,

[$_{TP}$ [$_{DP}$ D[-pron, +masc]]$_j$ T$_{[present]}$ [$_{VP}$ t V [$_{DP}$ D[+pron, +masc]]]]$_j$

b. John$_i$ likes himself$_i$ [+ pron, +masc]$_i$, and Peter$_j$ does [$_{VP}$ t V [$_{DP}$ D[+pron, +masc]]$_i$

However, in the case of the strict reading, when borrowing the lexical material from the antecedent in order to interpret the gap, the reflexive *himself* in the antecedent will need to be realized as a pronoun in the gap. This should not be a problem if it is the case that reflexives are composed of two parts: (i) the pronominal part *him* and (ii) the grammar formative *self* which is added to the first, if Principe A of the Binding Theory applies. Both the reflexive in the antecedent and the pronoun in the gap are occurrences of the same token in the Numeration, to be precise, $D_{[+pron, +masc]}$. In the first case it is lexicalized as *him* + *self*—since Principle A applies—but in the gap it will only be *him*.

Consider example (77), where there is a gender conflict between the antecedent and the gap. For the strict reading in (78), as in the case of the previous example, no feature value needs to be changed—although the reflexive *himself* will also need to be realized as a pronoun *him* in this example. For the sloppy reading (in (78)), however, the gender feature value needs to be changed: the masculine reflexive in the antecedent needs to be realized as a feminine reflexive in the gap. Also, when borrowing the lexical items from the antecedent to interpret the gap at LF, the reflexive *himself* needs to be changed into *herself* in the gap. Both reflexives are occurrences of one token ($D_{[+pron, +masc]}$), but the gender feature value in the elision site is changed to feminine, therefore, it is lexicalized as *himself* in the antecedent, but the corresponding word for that category in the gap is *herself*.

(77) John likes himself, and Susan does too.

(78) a. John$_i$ likes himself$_i$ [+ pron, +masc]$_i$, and Susan$_j$ does [$_{VP}$ t V [$_{DP}$ D[+pron, +masc]]$_i$

b. [$_{TP}$ [$_{DP}$ D[-pron, +masc]]$_i$ T$_{[present]}$ [$_{VP}$ t V [$_{DP}$ D[+pron, **+masc**]]]]$_i$,
[$_{TP}$ [$_{DP}$ D[-pron, -masc]]$_j$ T$_{[present]}$ [$_{VP}$ t V [$_{DP}$ D[+pron, **-masc**]]]]$_j$

A similar situation is found in the case of sentence (79). No feature value change is needed for the strict reading, but the gender feature needs to be changed in the gap in the case of the sloppy reading (see readings in (80) and (80) respectively). As in the previous example, in the case of the sloppy interpretation, both reflexives are occurrences of the same token in the Numeration; one occurrence of that token in the antecedent is realized as *herself* , while the corresponding lexical item in the gap is *himself* :

(79) Susan likes herself, and John does too.

(80) a. [$_{TP}$ [$_{DP}$ D[-pron, -masc]]$_i$ T$_{[present]}$ [$_{VP}$ t V [$_{DP}$ D[+pron, **-masc**]]]]$_i$,
[$_{TP}$ [$_{DP}$ D[-pron, +masc]]$_j$ T$_{[present]}$ [$_{VP}$ t V [$_{DP}$ D[+pron, **+masc**]]]]$_j$

b. Susan$_i$ likes himself$_i$ [+ pron, -masc]$_i$, and John$_j$ does [$_{VP}$ t V [$_{DP}$ D[+pron, +masc]]$_j$

We can explain the sentences in (72), (73), and (74) above in the same way (we repeat these below). Consider sentence (81) first. The strict reading does not involve any feature change, and the lexical item in the antecedent *my* is copied into the gap without any modification. For the sloppy reading, both the value of the number and person features need to be changed from 1st person singular in the antecedent to 3rd person plural in the gap. At the same time these occurrences of the pronoun are realized as *my* in the antecedent, but the corresponding form in the gap is *their*:

(81) I turned in my assignment, but most of the students didn't ~~turn in their assignments~~.

(82) a. $[_{TP}$ $[_{DP}$ D[+pron, 1st, sing]]$_i$ … $[_{DP}$ D[+pron, **1st**, **sing**]]]$_i$,
$[_{TP}$ $[_{DP}$ D[+pron, 3rd, pl]]$_j$ … $[_{DP}$ D[+pron, **3rd**, **pl**]]]$_j$

For the sentence below the feature that needs to be changed is [pronoun], in the antecedent its value is [-pronoun], while in the gap it is [+pronoun]:

(83) Mary loves John$_i$, and he$_i$ thinks that Sally does ~~love him$_i$~~ too

(84) Mary loves John$_i$, and he$_i$ thinks that Sally does $[_{VP}$ t V $[_{DP}$ D[+pron, +masc]]$_i$

The same kind of reasoning applies to sentence (85) below, where the feature [pronoun] also needs to change value. In this case, *John* will be realized in the gap as *himself*, since in this case Principle A is at work:

(85) I shaved John$_i$, because he$_i$ wouldn't ~~shave himself~~$_i$.

(86) I shaved John$_i$, because he$_i$ wouldn't [$_{VP}$ t V [$_{DP}$ D[+pron, +masc]]$_i$

In the case of these last two sentences, the question of how two occurrences of the same token can be lexicalized in a different way (to be precise, as *John* and *him*, and as *John* and *himself* in (83) and (85) respectively) can be raised, but it doesn't look like an impossible state of affairs.

To close up the discussion in this section, consider the examples that follow and the contrast with respect to the available readings. Compare examples (87) and (89), and (91) and (93):

(87) John saw his mother, and Mary did too.

(88) a. John saw his mother, and Mary saw his mother.

b. John saw his mother and Mary saw her mother.

(89) John saw the bastard's mother and Mary did too.

(90) a. John saw his mother and Mary saw his mother.

b. *John saw his mother and Mary saw her mother.

(her = the bitch's mother)

(91) I visited my mother, and they did too.

(92) a. I visited my mother, and they visited my mother.

b. I visited my mother, and they visited their mother.

(93) I visited Lisa's mother, and they did too.

(94) a. I visited Lisa's mother, and they visited Lisa's mother too.

b. *I visited Lisa's mother and they visited Susan's mother.

Based on these examples and on the rest of examples that have been discussed, one can conclude that there is a division between lexical items such as *his, her*, *himself* and *him*, and lexical items such as *Lisa, Susan*, *see* and *eat*. In other words, there is a distinction between the functional lexicon and the substantive lexicon (or between closed and open class lexical items) with respect to what counts as an occurrence of a token from the Numeration in ellipsis contexts. Functional lexicon alternations (e.g. *his/her* or *himself/him*) count as identical for ellipsis—they are occurrences of the same token in the numeration—while the case with substantive lexicon alternations (e.g. *Lisa/Susan*) is different. These last ones are not consider identical under ellipsis, i.e., they do not count as occurrences of the same token in the numeration. It will be interesting to investigate the extent of this generalization with more examples, but I will not pursue the matter here.

3.3.4 Summary

I have addressed the question of whether a condition on the identity of syntactic structure is respected in ellipsis by considering certain sentences that superficially seem to argue against such a restriction. I have argued that identity is met in ellipsis context, but that a more abstract notion of identity is needed. I have proposed identity of syntactic categories as the relevant condition for ellipsis to apply. Based on the proposal sketched in section 3.2 and on this new definition of identity I have offered, a unified account for several cases of partial syntactic identity can be advanced. The differences that can be found between antecedent and elision in terms of (i) verbal morphology, (ii) agreement features (person, number, and gender), and (ii) nominal expressions are grouped under identity of categories.

3.4 Conclusion

By assuming that lexical insertion is a late process in the derivation I can not only analyze ellipsis as a null lexicalization process (rather than deletion or interpretation), but also (i) account for strict and sloppy readings in a derivational manner—sloppy and strict readings emerge at different stages in the derivation, before and after lexical insertion respectively—and (ii) give an answer to the identity restrictions on ellipsis—identity of syntactic structure is the relevant notion and it is met before lexical items are part of the derivation.

3.5 Appendix

An issue that can be raised for the proposal presented here is whether it can account for ellipsis in contexts other than coordination. VPE ellipsis also takes place in subordinate clauses, as examples (95) and (96) show—an example of an ellipsis adjunct clause, and of Antecedent Contained Deletion (ACDs) respectively:

(95) Mary visited his mother before Susan did.

(96) I saw all the movies that you did.

In the case of coordination, I have said that the antecedent VP is built first and then an iteration rule builds the second VP in a parallel plane—I justified this move on the fact that coordinators subcategorize for two or more elements. However, this proposal does not seem to be extendable to subordination contexts. There are two questions in this sense: (i) How exactly is the derivation carried out? Is the antecedent or the gap built first? This is an important issue because we have said that null lexicalization applies to those VPs that are built by the iteration rule, or in other words, after the first built VP which functions as antecedent. And (ii) Why should an iteration rule apply and build a second VP, when subordinates do not subcategorize for two or more elements? I address the first question here. There is not much that I can say about the motivation for a rule of iteration to apply in subordinate contexts at this point.

In the minimalist program, the general assumption is that derivations are carried out bottom-up (and right-to-left). So, in general terms, for a sentence like (97) the order in which the words are merged is the following: (i) *her* is merged with *class*, (ii) the resulting DP is merged with the verb *cancel*, and (iii) this is merged with the subject *Peter*.

(97) Peter cancelled her class.

If it is the case that derivations are carried out in that way, then it is not very clear how the antecedent can be built first in the case of subordination, especially in example (96), where the gap is embedded inside the antecedent. For an example like (95), however, since the elided VP is inside an adjunct clause, it could be the case that the elision site is built after the main clause, independently from it, and merged as an adjunct to the main clause at the end of the derivation. This possibility does not exist for the ACD example.

Nevertheless, in the past years another option that has been explored (eg. Philips (1996), Drury (1998) and Guimaraes (1999)) is that derivations are carried out top-down, parallel to the way that syntactic structure is computed on the processing side. The question of whether derivations are bottom-up or top-down is still open to debate. Let us assume for the sake of discussion that derivations on the competence side can in fact be top-down, then the issue of VPE and subordination can be given a preliminary answer. In the case of top-down derivations, both of the examples above do not represent a problem for the proposal discussed in this chapter. In both cases the antecedent is assembled first, followed by the elision site. This order in which both antecedent and gap are built determines which of the two VPs is lexicalized and which one is not.

It is not the purpose of this work to argue in one direction or another, but it should be noted that other possibilities with respect to sentence derivations can be explored, and that this can help us not only solve the problem mentioned for subordination, but also make derivations on the competence and the performance side look more alike—this is a desirable consequence, since it reinforces homogeneity in the system.

CHAPTER 4: RELATED WORK ON ELLIPSIS AND PARSING

In this chapter, I discuss the proposal of Lappin and McCord (1990), which claims that ellipsis resolution involves the reconstruction of the elided VP by associating it with the syntactic structure and lexical items of the antecedent VP, and that this resolution process takes place at the level of S-structure.

I start by addressing those cases that they offer in favor of syntactic reconstruction at the level of s-structure. I also offer a description of the algorithm, which I am going to assume in part for the proposal I present in the next chapter. Then, I consider some examples that their algorithm fails to account for, some of which argue in favor of ellipsis resolution at the LF level, rather than s-structure.

I also briefly review some of the cases that Hardt (1993)—which advances a semantic approach for VPE—uses to argue against syntactic approaches, and conclude that those potentially problematic cases can be accounted for in a syntactic reconstruction approach if the identity condition on syntactic structure is somehow relaxed.

These two proposals address two central questions: (i) whether the process of ellipsis interpretation is of a syntactic or a semantic nature, and (ii) the level at which this takes place (s-structure or LF); they also set the stage to introduce our algorithm in Chapter 5, which is based on the minimalist theory framework.

4.1 Lappin and McCord (1990): Anaphora Resolution in Slot Grammar.

Lappin and McCord (1990) present three algorithms to resolve anaphora in Slot Filler Grammars: (i) an s-structure algorithm to interpret elliptical VPs, (ii) a syntactic filter on pronominal reference, and (iii) an algorithm for anaphor binding. The output of the first constitutes the input for the other two. It is the algorithm in (i) that we are interested in, and that will be mainly discussed in later sections.

I start by introducing Slot Grammars and the slot filling strategy, which will provide the reader with the basic toolkit to follow the discussion on the ellipsis resolution algorithm. Two arguments in favor of ellipsis resolution at s-structure (subjacency effects and subdeletion sentences) are introduced then, followed by the definition of the algorithm itself, and some parse examples. The discussion is finished with some cases that this algorithm fails to account for.

4.1.1 Slot Grammars: An Overview of the Formalism.

Lappin and McCord (1990) present an algorithm for VPE resolution based on Slot Grammars formalism. Slot Grammars are lexicalist grammars, which are organized around the filling of the slots associated with the head word of a phrase. The processing of words is done left-to-right, but phrases grow middle-out: starting with the head and adjoining slots either to the left or to the right of the head.

Each phrase is built out of a head word and some slots, or modifiers, associated with the head. There are two types of slots: (i) complement slots—those specified in the lexical entry of a head word—and (ii) adjunct slots—specified by the grammar. Slots represent grammatical relations such as subject, object, indirect object, etc. Thus, building the structure of a sentence consists of satisfying the SlotFrame of the head of a phrase. For a sentence like that in (1), the head word is *love* and its SlotFrame will at least consist of the list (subj, obj), which must be satisfied. *John* will fill the subject slot, and *Mary* will fill the object slot.[1]

(1) John loves Mary

There are rules for slot filling. Slot filler rules give conditions not only on the filler phrase (i.e., type of phrase: NP, PP, etc.), but also on its relation to the higher phrase. Together with these, ordering rules are also specified, which state conditions on the position—left or right—of the slots with respect to the head (Slot/head rules), as well as with respect to each other (Slot/slot rules).

Slots are optional by default, but they can be specified as obligatory by either a lexical entry or by the grammar—due to characteristics of the current phrase, or of the higher phrase to which a slot is related. If a slot is obligatory it can be filled in the current phrase, but also in a raised position; or in other words, in a higher phrase. Consider (2) and (3):

[1] In this example, both the subject and object slots are filled by trivial phrases, but each slot may be a phrase with a head and a SlotFrame itself, and it would be satisfied in the same manner.

(2) [Which movie did John like *obj*]?

(3) [Which movie did Mary think [John liked *obj*]]?

In (2) the object slot of the verb *like* is filled in a raised position inside the same clause. In (3), however, the object slot of the same verb is filled in a raised position in the higher phrase. The position that this slot would have been adjoined if there had been no raising is marked with italics.

Consider sentence (4), an example of a sentence that includes a raised element (the wh-element *who*) discussed by Lappin and McCord, together with the output parse produced for the sentence in (5):

(4) Who did John say wanted to try to find him?

(5)	subj (n)	who (**X2**)	noun
	top	do1 (X1, X3, X4)	verb
	subj (n)	John (X3)	noun
	auxcmp (inf (bare))	say (X4, X3, X9, u)	verb
	obj (fin)	want (X9, **X2**, **X2**, X12)	verb
	preinf	preinf (X12)	preinf
	comp (en| inf| ing)	try (X12, **X2**, X13)	verb
	preinf	preinf (X13)	preinf
	obj (inf)	find (X13, **X2**, X14, u, u)	verb
	obj (fin)	he (X14)	noun

The information about the parse is displayed as follows. Each line represents a node in the tree. The first column indicates the slot filled by the node (i.e. the argument role of the category), the second column represents the sense name of the head word with a list of the category's arguments (the SlotFrame): the first marker is the marker variable for the node/phrase itself, and the rest are the marker variables for the argument slot filler phrases. In raise operations the complement arguments are unified with the marker variables of the filler complement phrase, in other words, unification of marker variables is carried out in the sentence; the marker variable of the phrase *who* (X2) is unified with the subject variables of *want*, *try*, and *find*.[2]

4.1.2 An S-Structure Algorithm for VPE Interpretation

4.1.2.1 Motivation for VPE Resolution at S-Structure: Subjacency and Subdeletion.

Lappin and McCord argue in favor of a S-structure treatment of VPE phenomena within a slot filler framework. They give two reasons for such a move: (i) subjacency effects in ACDs, and (ii) subdeletion cases of ACDs.

They argue against May (1985), who advances a LF-based approach to VPE resolution. According to him, VPE interpretation for ACDs need to take place at LF, only after Quantifier Raising (QR) has applied, in order to avoid the interpretive

[2] I have marked the unified variables with bold in the example. With this process of unification deep grammatical roles are obtained while parsing, through the unification of complement marker variables with the variables in the argument frames of their heads.

regress that will result from copying the matrix VP (which includes the elided VP) into the elided VP. To a sentence like (6), QR applies as in (7), and then the matrix VP is copied into the elided VP at LF as in (8). The VP copying rule, thus, applies at LF.

(6) Dulles suspected everyone who Angelton did.

(7) $[_{IP'}$ $[_{NP1}$ everyone who$_1$ Angleton did] $[_{IP}$ Dulles suspected t_1]

(8) $[_{IP'}$ $[_{NP1}$ everyone who$_1$ Angleton **suspected t_1**] $[_{IP}$ Dulles suspected t_1]

Lappin and McCord note that May's approach to ellipsis faces two problems. The first problem is that it cannot account for the subjacency effects shown in (9). Subjacency is a condition that constrains the binding of traces resulting from movement at S-structure. According to May's proposal, the elided VP is empty at S-structure, so it is not clear how an operator-trace chain can be established at this level.

(9) a. John read everything which Bill believes he did.

b. *John read everything which Bill believes the claim that he did.

An LF-based account also does not extend to all the cases of ACDs, such as subdeletion, where arguments or adjuncts of the partially empty VP are overtly realized:

(10) John writes more books than Bill does articles.

(11) John showed everything to Mary which he did to Bill.

(12) John reviewed the play for The New York Times shortly after Bill did for The Washington Post.

Lappin and McCord (1990) suggests that May's analysis cannot account for subdeletion cases, since it treats VP ellipsis as a relation between an empty VP and an antecedent VP.

They show how it is possible to capture the properties of ACDs in general if we assume that the elided VP is structured and may contain arguments or adjuncts: these may be lexically realized or they may be traces. Thus, a sentence like (13) has a structured VP as in (14), and it is interpreted as in (15), by copying the verb from the antecedent, avoiding the interpretive regress without the need to invoke LF:

(13) Dulles suspected everyone who Angleton did.

(14) Dulles suspected [$_{NP}$ [$_{N'}$ everyone who$_1$ Angelton did [$_{VP}$ [$_{V}$] [$_{NP}$ t$_1$]]]]]

(15) Dulles suspected [$_{NP}$ [$_{N'}$ everyone who$_1$ Angelton [$_{VP}$ [$_{V}$ **suspected**] [$_{NP}$ t$_1$]]]]]

According to them, the advantages of thinking about the VP in these terms are: (i) subjacency can be checked for at S-structure, (ii) we can handle ACDs and subdeletion examples with the same interpretation procedure.

4.1.2.2 The Algorithm

Lappin and McCord (1990) define VP anaphora as an s-structure relation between the head V and selected arguments and adjuncts of a structured empty (or partially empty) elliptical VP, and the head and adjuncts of an antecedent VP. The interpretation procedure copies the head V of the antecedent VP into the elided VP, and specifies which arguments and adjuncts of the antecedent are inherited by the elliptical VP, using a slot filler type of approach, as discussed in Section 3.1.1.
The algorithm that they present follows in (16). It consists of four main steps, which are divided in other sub procedures. Step A, describes the environments in which elliptical verb-antecedent pairs may be found. Step B, C, and D specify the steps for the interpretation procedure.

(16) "A. Identify an elliptical verb-antecedent verb pair <V,A> as follows.

1. An elliptical verb V is identified by the presence of an auxiliary verb or the infinitival complementizer "to", where the auxiliary verb or the complementizer does not have a realized verb complement.
2. A candidate A for an antecedent of V is a verb which is not elliptical and not an auxiliary verb with a realized complement.

3. Check that A and V stand in at least one of the following relations:

    a. V is contained in the clausal complement of a subordinate conjunction SC, and the SC-phrase is either (i) an adjunct of A, or (ii) and adjunct of a noun N and N heads an NP argument of A, or N heads the NP argument of an adjunct of A.[3]

    b. V is contained in a relative clause which modifies a head noun N, and either (i) N heads an NP argument of A, or (ii) N heads a complement of an adjunct of A.

    c. V is contained in the right conjunct of a sentential conjunction S, and A is contained in the left conjunct of S.

B. Generate a new tree in which A is substituted for V as the head of the elliptical verb phrase VP' which V heads, and A is assigned the agreement features required by the head of VP'. The new occurrence of A will be referred to as A'.

C. Consider each argument slot Slot_i in the argument frame of A.

[3] Here I include the definition of the relation Contain, that Lappin and McCord provide:

> (i) "We define the predicate P is contained in Q recursively as follows. A phrase P is inmediately contained in a head Q iff (i) P is an argument of Q, or (ii) P is an adjunct of Q. P is contained in Q iff (i) P is immediately contained in Q, or (ii) P is immediately contained in a head R, and (the phrase with head) R is contained in Q."
> ("Anaphora Resolution in Slot Grammars", page 11)

1. If Slot$_i$ is filled by a phrase C, then

   If there is a phrase C’ in VP’ which is of the appropriate type for filling Slot$_i$, then fill Slot$_i$ in the argument frame of A’ with the marker variable of C’.

   Else,

   Fill Slot$_i$ in A’ with the marker variable of C, and list C as a new argument of A'.

2. If Slot$_i$ is empty in the frame of A, it remains empty in the frame of A’.

D. For each adjunct Adj of A, if there is no adjunct of the same type as Adj in VP’, then list Adj as a new adjunct of A’.”

4.1.2.3 The Algorithm at Work: Some Examples

Now let us look at three examples to see how the algorithm actually works. Consider sentence (17), the ACD example I discussed above, and the output parse in (18):

(17) Dulles suspected everyone who Angelton did.

(18) a. Parse before VP Anaphora Algorithm

subject	Dulles (X4)	noun
top	suspect (X1, X4, **X5**)	verb
obj	everyone (**X5**)	noun
obj	who (**X5**)	noun

subject	Angelton (X9)	noun
nrel	do (X7, X9, **X5**)	verb

b. Parse after VP Anaphora Algorithm

Antecedent verb-elliptical verb pairs: suspect.2 – do.6

Elliptical verb-new argument pairs: none

Elliptical verb-new adjunct pairs: none

Subj	Dulles (X4)	noun
top	suspect (X1, X4, **X5**)	verb
obj	everyone (**X5**)	noun
obj	who (**X5**)	noun
subj	Angelton (X9)	noun
nrel	**suspect** (X7, X9, **X5**)	verb

In sentence (17), only the verb from the antecedent VP is copied into the elided VP; no arguments or adjuncts are inherited from the antecedent. The object in the elision site is successfully recognized to be a trace: we can observe the unification of the variable X5 in the object slot with the phrase marker of the head of the relative clause *everyone*, and with that of the wh-phrase *who*. Consider (19) now, and the output parse in (20):

(19) John wrote notes to everyone who asked him to.

(20) a. Parse before VP Anaphora Algorithm

subj	John (X3)	noun
top	write (X1, X3, X4, **X5**, u)	verb
obj	note (X4)	noun
iobj	to (X7, **X5**)	prep
objprep	everyone (**X5**)	noun
subj	who (**X5**)	noun
nrel	ask (X9, **X5**, X11, X12)	verb
obj	he (X11)	noun
comp	preinf (X12)	preinf

b. Parse after VP Anaphora Algorithm

Antecedent verb-elliptical verb pairs: write.2 – preinf.9

Elliptical verb-new argument pairs: preinf.9 – note.3

Elliptical verb-new adjunct pairs: none

subj	John (X3)	noun
top	write (X1, X3, X4, **X5**, u)	verb
obj	note (X4)	noun
iobj	to (X7, **X5**)	prep
objprep	everyone (**X5**)	noun
subj	who (**X5**)	noun
nrel	ask (X9, **X5**, X11, X12)	verb
obj	he (X11)	noun

comp	preinf (X12)	preinf
auxcomp	**write1 (X15, X11, X4, X5)**	verb

In sentence (19) the verb is copied from the antecedent into the ellipsis site. The direct object of *write1* (X4) is inherited from the antecedent, while the indirect argument (X5) is realized as a trace bound by the wh-phrase (see unification of marker variables).

Consider (21) now, an example of subdeletion. This subdeletion sentence is resolved in the same manner as the other ellipsis examples. This is an advantage of this algorithm (which assumes that the elided VP is not structurally empty). Observe the resulting tree after VP anaphora interpretation, which is displayed in a linear format under (22):

(21) Max writes more letters to Sam than Mary does to Bill.

(22) Antecedent verb-elliptical verb pairs: write.2 – do.9

Elliptical verb-new argument pairs: write.9 – letter.4

Elliptical verb-new adjunct pairs: none

Max (X3) write (X1, X3, X4, X5, u) more (X13) letter (X4) to (X11, X5) Sam (X5) than (X1, X7) Mary (X8) **write** (X7, X8, **X4,** X9**)** to (X7, X9) Bill (X9)

In order to interpret the elision in (21), the verb from the antecedent VP is copied into the elided verb phrase, the direct object (X4) is inherited from the antecedent too, and the indirect object (X9) is that one overtly realized in the elided VP (*to Bill*). Compare how the direct and indirect objects are represented—marked with the solid and the dashed arrows, respectively. The former is only listed in the arguments list (or SlotFrame); represented by the marker variable X4, as the inherited argument that it is. The latter, however, is part of the argument list (represented in the SlotFrame by the variable X9), but it is also listed as a member of the structure of the tree (Bill (X9))—in other words, it is structurally realized in the lower/elided VP.

I would like to call the reader's attention to the following fact: even though the elided VP inherits the direct object, its syntactic structure is not copied into the gap. The direct object exists in the elided VP SlotFrame by means of a marker variable, or in other words, a pointer/an address that makes reference to the syntactic structure realized in the antecedent clause. Thus, their proposal does not face a computational complexity problem, since the number of output symbols with respect to input symbols does not grow unboundedly because of copy of the syntactic structure. I will discuss this issue in more detail in Chapter 5, section 5.1.1**.**

4.1.3 Some Problematic Examples for this Algorithm

4.1.3.1 Strict and Sloppy Readings

When Lappin and McCord's algorithm reconstructs the elided VP, it does so in a way that does not always permit us to get all the possible readings for the elided VP. However, it has been shown in the literature that there are ellipsis sentences for which more than one reading is available, i.e. where the elided VP receives either a strict or sloppy interpretation. Consider sentence (23) below, together with the readings in (24) (I include one example here for illustrative purposes, but I refer the reader to Chapter 2, section 2.3, where this matter is discussed in detail):

(23) John saw his mother and Peter did too.

(24) a. John$_1$ saw his$_1$ mother, and Peter$_2$ saw his$_1$ mother.

b. John$_1$ saw his$_1$ mother, and Peter$_2$ saw his$_2$ mother.

For the sentence in (23) there are two possible readings: (i) the strict reading in (24), where the pronoun in the elided clause is interpreted as referring to the subject in the antecedent, and (ii) the sloppy reading in (24), where the same pronoun is interpreted as referring to the subject in the elided clause. Let us see how Lappin and McCord account for strict and sloppy readings.

For the interpretation of pronouns and anaphors, Lappin and McCord propose an Anaphor Binding Algorithm and a Syntactic Filter on Pronominal Anaphora. These last two together cover more or less the same amount of data as the Binding

Theory does, however, they are both defined on configurations that apply to the SlotFrame of the verb—they do not depend on c-command relations like Principle A, for example, does.

The output of the VP resolution algorithm is the input for those two other routines.

Consider the sentence in (25), and the available readings in (26): the sloppy reading represented under (26), and the strict one under (26). For the sentence in (25) Lappin and McCord's algorithm predicts only one reading: the sloppy one. Their interpretive mechanism copies the verb from the antecedent into the elided VP, and this copied verb inherits the anaphoric element *herself*, which is interpreted in the lower VP. The anaphor is bound by the subject in the second clause and the sloppy reading comes out. The output of their parsing algorithm is included in (27) below:

(25) The girl will write a book about herself, and Mary might too.

(26) a. The girl … about herself, and Mary … about herself too.

b. The girl … about herself, and Mary … about her too.

(27) Antecedent verb-elliptical verb pairs: write.4 – may.12

Elliptical verb-new argument pairs: may.12 – book.6, may.12 – about.7

Elliptical verb-new adjunct pairs: none

the (X11) girl (X9) will (X8, X9, X10) write (X10, X9, X12, u, u) a (X15) book (X12) about (X12, X16) herself (X16) and (X1, X8, X18) Mary (X19) may (X8, X19, u) write (X24, X19, X12) too (X18)

Antecedent NP-reflexive pairs:[4]

girl.2 – herself.8, Mary.11 – herself.8

Evidently, a descriptively adequate algorithm should also predict the strict reading. This is a flaw that Lappin and McCord themselves mention. They observe how in order to obtain the strict interpretation, according to which *herself* is bound by *the girl*, it would be necessary to refine the VP anaphora algorithm to permit it to identify inherited arguments by abstract referential indices, such as phrase marker variables, as well as by the heads of the argument phrase.

They just discuss the example above, but we can observe how in other sentences the number and the kind of readings that are predicted by their algorithm vary. Consider sentence (28) now. The interpretative mechanism would copy the verb *talk* to the elided VP, and this copied verb would inherit the pronoun *his*, which is interpreted in the lower VP. In this case, the readings that their algorithm predicts are two: strict and sloppy. The filter on pronominal anaphora will correctly predict that

[4] The output of applying both the anaphor binding algorithm and the filter on syntactic coreference to sentences are pairs of coreferent antecedents and anaphors, or antecedents and pronouns respectively. This is what these pairs are showing, corefering elements.

the pronoun in the elided VP could co-refer either with the subject in the first clause, or with the subject in the second clause.

(28) Peter talked to his teacher, and Tom did too.

Nevertheless, for a sentence like (29) or (30) the algorithm makes the wrong predictions again. One of the conditions that must be respected for co-reference by the Anaphor Binding Algorithm and the Syntactic Filter on Pronominal Anaphora is the identity of agreement features between antecedent and anaphor/pronoun. If the antecedent and the inherited pronoun/anaphor have incompatible agreement features (e.g. *Mary* and *his* in (29); or *Susan* and *himself* in (30)), then they cannot co-refer. However, as we saw in Chapter 2, section 2.2 (where Kitagawa's analysis is discussed), gender conflicts do not preclude coreference in all the examples.

(29) Peter talked to his teacher, and Mary did too.

(30) John talked about himself, but Susan didn't.

Sentence (29) will only be assigned the strict reading (that one in which the pronoun in the elision site is bound by the subject in the first conjunct). The sloppy reading (pronoun bound by subject in second conjunct) is precluded, since there is an agreement mismatch between *his* and *Mary*. Sentence (30) cannot be assigned any reading, the strict reading does not obtain for the same reasons as in (25) above. And the sloppy interpretation (the only one that could be available, since *himself* is inherited by the elided VP) is not acceptable due to agreement features mismatch.

Thus, we can conclude that the strategy of inheriting the argument from the antecedent, and interpreting it (applying the anaphora/pronoun algorithms) in the SlotFrame of the elided VP does not yield the correct results. It seems that inheriting arguments in a more abstract manner is necessary for VP ellipsis interpretation. I will propose something to this regard in Chapter 5, section 5.3.2.

4.1.3.2 On the Necessity of the LF level

Fiengo and May (1994) argue in favor of the need to reconstruct elliptical ACD sentences at the level of LF. They show that there are interactions between Binding Theory and Quantifier Raising (QR), which can only be accounted for at the LF level.

They analyze several ACD examples, and the level at which Binding Theory seems to apply for these. They observe how some of those cases are problematic for a syntactic approach to VPE where the elided VP is fully interpreted at the level of s-structure, such as that defended by Lappin and McCord (1990)—in order to interpret pronouns and anaphors the reconstructed VP is considered at the level of s-structure, not later on. There are sentences like (31), however, which cannot be analyzed at s-structure. As Fiengo and May observe, s-structure and LF are in a feeding/bleeding relationship with respect to Binding Theory, and all those cases where binding violations are bled from one level to the other would be unaccounted for (in this and in all the examples included in this subsection we are interested in those readings where the pronouns are interpreted anaphorically):

(31) Mary introduced him$_1$ to everyone that John$_1$'s mother wanted her to.

Sentence (31) violates Principle C at s-structure under this anaphoric reading, since the pronoun c-commands the name *John*. However, it is still grammatical. A look at its LF form in (32) provides the answer to why this sentence is accepted under the reading in (31).

(32) everyone that John$_1$'s mother wanted her to **[$_{VP}$ introduced him$_1$ to t]**
[Mary introduced him$_1$ to]

After QR has applied, and the elided VP (boldfaced) is reconstructed, the violation that existed at s-structure disappears. At LF the NP *John* is no longer c-commanded by the pronoun, it is free, and the Principle C violation is no longer sustained. Lappin and McCord's s-structure proposal cannot explain cases like this, where a violation is bled form s-structure to LF.

Consider the sentences in (33) and (34) now. Both of these sentences violate Principle C at s-structure, since in both cases the pronoun c-commands its antecedent, the name *John*. However, (33) is grammatical, while (34) is not. Once again, the correct distinction is made at the LF level (see (35) and (36)):

(33) Mary introduced him$_1$ to everyone that John$_1$ wanted her to.

(34) * Mary introduced him$_1$ to everyone that she wanted John$_1$ to.

(35) everyone that John$_1$ [$_{VP}$ wanted her to **introduce him$_1$ to t**] [Mary introduced him$_1$ to]

(36) *everyone that she [$_{VP}$ wanted John$_1$ to **introduce him$_1$ to t**] [Mary introduced him$_1$ to]

At LF (35) conforms to Binding Theory requirements. That is not the case with (36), which violates Principle B at LF, since the reconstructed occurrence of the pronoun is not locally free. In this last case, s-structure and LF stand both in a bleeding and a feeding relationship. The violation of Principle C at s-structure is bled, since the pronoun does not c-command the name at LF any longer. However, QR and VP reconstruction feed Principle B, because the reconstructed pronoun is c-commanded by the referential expression *John*.

In sum, an s-structure algorithm like that of Lappin and McCord's, despite otherwise wide coverage, faces some problems. The elided VP is not only reconstructed, but also interpreted at that s-structure. Binding Theory applies at this level too, and this is problematic because it does not cover those examples where BT interacts with QR. In Chapter 5, I present an algorithm that does not face this problem; it allows reconstructing some of the structure of the elided VP as the sentence is parsed, but it finishes interpreting the elision by applying binding theory to the output of LF-related operations, such as QR.

4.1.3.3 Some Remaining Issues

Another problem that the algorithm proposed by Lappin and McCord (1990) faces is the fact that VP anaphora resolution is totally dependent on accessing the antecedent. The SlotFrame of the antecedent is used to interpret the elided verb phrase. However, there are VPE examples where the antecedent does not precede the gap like (37) and (38) below:

(37) Although Mary didn’t, Peter went to the party.

(38) Mary did, but Peter did not pass the exam.

Examples like (37) and (38) above, where the relation is gap-antecedent instead of antecedent-gap, are unaccounted for. Lappin and McCord’s algorithm is dependent on accessing the antecedent to interpret the gap. It is not clear how the parse for this type of sentences will be resolved; either the elided verb phrase will not receive any interpretation since the antecedent is not available at the point where the gap is processed, or the gap will be assigned the wrong interpretation by accessing a predicate that precedes the gap, but which is not the antecedent.

Finally, consider (39)-(41) below. These sentences are examples of gapping—another type of verbal ellipsis, where the auxiliary, verb, and optionally some of the arguments (example (40)) and/or adjuncts (example (41)) are elided:

(39) Peter ate pasta, and Mary pizza.

(40) John talked with this boss about a salary raise, and Sam about a permanent contract.

(41) Max sometimes mistreats his friends, and Peter his mother.

The algorithm defined by Lappin and McCord above does not extend to this kind of elliptical structure. Step A of the algorithm—which is in charge of detecting the elision site and relating it to the antecedent—will not detect these gaps; since an auxiliary is not present and the detection routine is anchored to the existence of an auxiliary without a following verb.

One could imagine a solution to this problem, which is modifying Step A in order to cover those examples above too. Nevertheless, as it stands it remains a problem for two reasons: (i) one of the advantages of Lappin and McCord's proposal (which the authors offer as an argument in favor of their ellipsis account) is that different ellipsis phenomena are resolved with the same algorithm; but this strength becomes a weakness in the case of gapping. And (ii) gapping has been analyzed as a sentence grammar process (as opposed to VPE, which has been proposed to be a discourse type of phenomena); thus, an s-structure algorithm such as Lappin and

McCord's should arguably include the elliptical sentence grammar process par excellence.

In the next chapter, I will introduce a minimalist algorithm for VP anaphora which: (i) can build some structure for the elided verb phrase without resorting to the antecedent, so it can also account for cases where the relevant relation is gap-antecedent rather than antecedent-gap, (ii) covers VPE, ACD and pseudogapping cases, as well as the gapping cases, and (iii) offers an explanation for why gapping sentences (as opposed to other elliptical processes) display locality effects—i.e. there cannot be any material intervening between the elided clause and the antecedent, as the examples in (42) and (43) show:

(42) *Peter ate pasta, and **I think that** Mary pizza.

(43) *My best friend loves movies, and **everyone knows that** Susan plays.

4.1.4 Summary

Lappin and McCord (1990) present a VP anaphora resolution algorithm that applies at the level of s-structure. VP ellipsis is interpreted by resorting to the SlotFrame of the antecedent: the verb is copied from the antecedent into the elision site, but the arguments or adjuncts of that verb are inherited. This mechanism allows (i) treating different elliptical constructions (VPE, ACDs, and subdeletion) in a unified way, and also (ii) parsing ellipsis in an efficient way, since the syntactic structure of elided elements does not need to be copied into the gap: elided elements are inherited. I assume a similar mechanism in my proposal in next chapter (see Chapter 5, section 5.2.6.1).

However, we have also seen how there are certain facts about elliptical constructions that an s-structure algorithm fails to encompass; to be precise, cases where interaction between binding theory possibilities and quantifier raising at LF is observed, and which argue in favor of resolving ellipsis at LF. Thus, we find contradictory requirements, which seem to argue in the direction of a two-fold process of ellipsis resolution in which some of the work is done before reaching the LF level, and some at the LF level.

I have also mentioned other problematic cases posed by the combination of strict and sloppy readings, by examples where the antecedent precedes the gap, and by gapping sentences. In Chapter 5, I offer an alternative algorithm that tries to address all these issues.

4.2 Hardt (1993). Verb Phrase Ellipsis: Form, Meaning, and Processing.

Hardt (1993) claims that ellipsis resolution is done by the identification of a VP meaning at the discourse level. He presents the Proverb Theory, which treats VP-anaphora and NP-anaphora uniformly. He analyzes VPE as a proform, i.e. it has no syntactic structure, and it is interpreted by being semantically identified with its antecedent.

I do not get into a description of his proposal here—since it is not relevant to what I am going to present in the next chapter—but I rather have a quick look at the arguments that he offers against syntactic approaches to VP ellipsis, and in favor of a semantic approach, to see whether they are valid.

4.2.1 Binding and Island Constraints Do Not Apply to VP Ellipsis

Hardt (1993) claims that the two main arguments that have been offered in the literature in favor of syntactic reconstruction approaches do not go through. Those arguments are: (i) binding constraints, and (ii) island constraints, which seem to be at work in the gap site and argue in favor of an elided VP with syntactic structure.

Hardt (1993) also notes that Binding constraints are not always respected under ellipsis. Both in sentence (44) and (45) binding constraints do not seem to apply as they do with overt material. If syntactic material is copied from the antecedent into the elision the resulting sentences would be those in (46) and (47), which violate Principle C and Principle A of the Binding Theory, respectively:

(44) Harry got to Sue$_i$'s apartment before she$_i$ did.

(45) John$_i$ defended himself$_i$ better than his$_i$ lawyer could have.

(46) *Harry got to Sue$_i$'s apartment before she$_i$ did [go to Sue$_i$'s apartment].

(47) *John$_i$ defended himself$_i$ better than his$_i$ lawyer could have [defended himself$_i$]

Hardt observes that examples like (44) and (45), above oblige to relax the identity condition on ellipsis. He notes how Fiengo and May (1992) vehicle change theory—which allows for certain differences between the antecedent and the gap—accounts for Principle A and Principle C violations. The reconstructed VPs will look like in (48) and (49), after vehicle change applies, respecting binding theory constraints:

(48) Harry got to Sue$_i$'s apartment before she$_i$ did [go to her$_i$ apartment].

(49) John$_i$ defended himself$_i$ better than his$_i$ lawyer could have [defended him$_i$]

One of the differences that vehicle change allows for is the conversion of a name into a pronoun, as in (48). Another difference is the conversion of a reflexive into a pronoun (as in (49)). Since reflexives are composed of two parts (an anaphoric and a pronominal part), if only the pronominal part is copied into the elision, Principle A is respected. Nevertheless, Hardt argues that there are also Principle B violations under ellipsis, for which Fiengo and May's theory does not account. An example of this type of violation is sentence (50):

(50) A: Why do you want him$_i$ to play chess?

B: I don't, he$_i$ does [want him$_i$ to play chess].

However, we have seen in Chapter 2 (section 2.3), where I discuss Fiengo and May's (1994) proposal, that reconstruction is a symmetrical relationship. Consequently, in the same way that a reflexive can be reconstructed as a pronoun, a pronoun can be reconstructed as a reflexive, thus, respecting binding theory and supporting identity of structures.

With respect to the second argument, the island constraints, he claims that all those sentences in ((51)-(54)) which have been used to argue in favor of syntactic reconstruction for the elision site (there is a trace in the gap that needs to be bound; thus, requiring syntactic reconstruction (Chao 1987; Haik 1987; Tancredi 1992)) are examples of pesudogapping.

(51) John knows who$_k$ Bill criticized e$_k$, and Mary knows who$_i$ Sue did e$_i$.

(52) John read everything which$_i$ Bill did e$_i$.

(53) *John read everything which$_i$ Bill believes the claim that he did e$_i$.

(54) *John met everyone Op$_i$ that Peter wondered when he could e$_i$.

I believe that characterizing these sentences as pseudogapping just puts the problem aside, it does not solve it. If it was true that they are cases of pseudogapping, then how is pseudogapping treated? There is still a trace that needs to be bound in the elision site in those cases, and the Proverb Theory cannot account for them. I believe that elliptical constructions such as VPE and Pseudogapping can be and should be treated uniformly. In the next chapter, I propose a unified account for both elliptical constructions.

4.2.2 Some Problematic Cases for Syntactic Identity

Hardt discusses some examples that he considers to be problematic for a syntactic approach to VPE; we can divide those in two groups: cases (i) where syntactic identity between the antecedent and the elided VP seems not to be respected, or (ii) where the antecedent is not in a local relation with the gap.

With respect to the first group there are examples that involve reflexives (in (55)), negative polarity items (in (56)), and active/passive mismatches (in (57), (58) and (59)). Hardt claims all the examples below argue against identity of syntactic structure. In (55) the antecedent is *defended himself*, while the gap must be *defend him*, so as to conform with BT. In sentence (56) below the material in the elided VP should be *have some* rather than *have any*. The same kind of problem arises with Active/Passive mismatches between the antecedent and the gap, or vice versa:

(55) John defended himself, because his lawyer couldn't.

(56) Tom doesn't have any paper. Harry does, though.

(57) The information could have been released by Gorbachov, but he chose not to.

(58) A lot of this material can be presented in a fairly informal and accessible fashion, and often I do.

(59) Max fired Harry, although it was Tom who should have been.

Another situation in which Hardt questions the reality of syntactic identity is sentences where syntactic constraints on variables within the antecedent do not apply in the ellipsis site. The sentences in (60) and (61) both have a variable in the antecedent bound by a wh-operator. In both, the antecedent VP is *visit e*. If this syntactic representation was copied into the gap, the wh-trace *e* will not be bound in the elision, resulting in an ungrammatical sentence:

(60) China is a country that Joe wants to visit **e**, and he will too, if he gets enough money.

(61) China is one of the countries that Joe doesn't want to visit **e**. In the case of India, he does.

However, we have seen in Chapter 2 that a theory like that of vehicle change can account for all these cases, while at the same time maintaining the identity condition for ellipsis.

The other kind of examples that Hardt offers are cases where the relation between gap and antecedent is not local, cases that involve long distance antecedents (example (62)), and missing antecedents (examples (63) and (64)):

(62) I disagree with the writer who says funeral services should **be government-controlled**. The funeral for my husband was just what I wanted, and I paid a fair price, far less than I had expected to pay. But the hospitals and doctors should.

(63) I will if you will.

(64) You shouldn't have.

These examples are not problematic for the analysis that I propose for VPE in the next section. As we are going to see there, a VP is predicted without the need to access the antecedent. This predicted VP is interpreted at LF, where there is access to c-commanding material and to not c-commanding material as well.

One can conclude that the two arguments (binding and island constraints) offered in the literature in favor of syntactic reconstruction for ellipsis can still be sustained.

4.3 Closing Remarks

I have discussed Lappin and McCord's (1990) algorithm for VPE resolution, which is representative of the syntactic view on ellipsis. From the discussion of this proposal, we can conclude that it seems necessary to claim that the elided VP has syntactic structure if we want to offer (i) a unified treatment of different elliptical constructions—account for VPE, subdeletion or pseudogapping, as well as examples of VPE with traces that need to be bound inside the elided VP.

We have also seen that the arguments in favor of syntactic reconstruction (binding and island constraints effects) are effective. And also, that those examples that seem to question the validity of a syntactic approach are no longer relevant if the identity condition on ellipsis is somehow relaxed, along the lines of the theory of vehicle change proposed by Fiengo and May (1994), for example.

CHAPTER 5: A PROCESSING ACCOUNT OF LOCALITY EFFECTS IN VERBAL ELLIPSIS

In this chapter, I discuss verbal ellipsis from a processing point of view. The question I want to answer is how elliptical sentences are parsed. How are the sentences in ((1)-(3)) processed? What about the elision site? We saw in Chapter 3 that our competence theory argued in favor of having a structured elision. Now, the question is what kind of structure is assigned to these sentences when they are processed.

(1) Lisa went to the theater, and you did too.

(2) John sent a packet to Susan, and Peter did to Mary.

(3) Bill likes jazz, and Tom opera.

The problem for parsing is (i) to detect the gap, and (ii) to resolve/interpret it. As a preview of what is coming, I would like to mention here that there is a contrast between VPE and pseudogapping constructions on the one hand, and gapping on the other. For the first two, the gap is detected by the presence of an auxiliary—in (1) and in (2) the auxiliary *did* signals the gap, and predicts a VP—the antecedent only needs to be accessed to interpret the predicted VP—while in (3) the antecedent needs to be consulted to assign structure to the gap and for interpretation purposes.

I also address well-known locality restrictions that affect gapping constructions, but do not apply to VPE or pseudogapping cases. I offer an explanation for these based on (i) the difference among elliptical constructions mentioned in the previous paragraph, (ii) low initial attachment of coordinates, and (iii) spell-out operations which render syntactic material unavailable. The last two together determine when left-context, i.e. the antecedent in ellipsis cases, is available.

In what follows, I offer an analysis for parsing different verbal ellipsis constructions with an algorithm based on the minimalist operations (merge, move, spell-out), which takes into account efficiency and economy considerations, and which makes use of local information.

I present a two-fold process for ellipsis resolution in which part of the work is done on-line and part at LF. The parser computes certain syntactic structure for the gap (we will see how much in the sections that follow) on-line with the use of local information. At LF other operations, such as Quantifier Raising take place.

5.1 What Is the Structure of the Elision Site?

Before I define how the resolution of ellipsis proceeds, I would like to briefly consider what the structure of the elided constituent looks like. As we saw in Chapter 1, the elision may be taken to have syntactic structure or to be empty. I review the arguments in favor of having a structured elision site.

It has been argued (e.g. Chao (1987), Haik (1987)) that the elided verb phrase in VPE examples is not structurally empty. They have shown that there must be syntactic structure in the elision site, since there are examples of ellipsis where traces inside the elided VP need to be bound, and also examples where subjacency seems to be at play:

(4) John knows who$_i$ Bill criticized t$_i$, and Mary knows who$_k$ Sue did t$_k$.

(5) Tom read everything which$_i$ Bill did t$_i$.

(6) *Tom read everything which$_i$ Bill believes the claim that he did t$_i$.

(7) *John met everyone Op$_i$ that Peter wondered when he could t$_i$.

In sentence (4) and (5), there is a trace inside the verb phrase that needs to be bound. Sentence (6) is ungrammatical because the trace that resides inside the VP cannot be bound from the position where the wh-element is. The same reasoning applies to sentence (7); the operator is too far away from the trace to bind it.

Lappin and McCord (1990) offer ACDs and subdeletion cases of ACDs as an argument in favor of having a structured VP. In the case of ACDs having a structured VP with empty positions or positions filled with a trace avoids the infinite regress that interpreting the gap at s-structure will cause. For ACD subdeletion cases, the verb phrase needs to be structured in order to attach the overt object:

(8) Dulles suspected everyone who Angleton did.

(9) John writes more books than Bill does articles.

Pseudogapping sentences, where one of the arguments or adjuncts are remnants of the elision are also another case which support the claim of having a structured VP:

(10) John sent a packet to Susan, and Peter did to Mary.

All these examples show that when parsing VPE, pseudogapping, and ACDs the elision site cannot just be assigned an empty category. There are traces that need to be bound, or overt elements (i.e. remnants of elision) that need to be attached to the syntactic structure that is being computed for the sentence, and this is not possible unless syntactic structure is built for the VP.

Now the question is how the structure of the elided verb-phrase is built, since there are no overt lexical items in the elision site from which a VP could be built bottom-up (i.e. no verb). One possibility is to copy the structure from the antecedent into the gap. However, if the same amount of structure that is built for the antecedent VP is copied for the elided VP, VP anaphora would not be resolved in a computationally efficient way—the number of output structure symbols would be much bigger than the number of input symbols, since the elided structure can (at least theoretically) grow infinitely. The elided VP can be structurally as simple as in (11), or as complex as in (12):

(11) Sam [saw that film], and Mary did too.

(12) Sam [said he thinks he has already seen that film that those friends of yours who love foreign movies will watch next weekend], and Peter did too.

The problem of computational complexity was noticed by Peters (1973). This motivates our proposal about ellipsis as a VP-structure sharing relation.

5.1.1 The Computational Complexity Problem: An Argument against Copy

Peters (1973) notices that one problem that an Aspects type of theory has to face is the fact that the mapping between deep structure and s-structure involves several copying and deletion operations. Assuming that the job of the parser is to recover deep structure from the surface structure string, then the number of symbols in the output representation grows exponentially with respect to the number of input symbols, which allows potentially non-recursively enumerable derivations.[1] Peters (1973) discusses an example of Equi-NP deletion in order to illustrate his argument on complexity of unbounded deletion. The structure that needs to be computed for sentence (13) is that one in (14):

(13) Their sitting down promises to steady the canoe.

[1] In the minimalist framework that I assume here, the level of deep structure does not exist. However, the complexity problem does not disappear, since the parser still needs to recover deleted material for interpretation purposes at LF.

(14) [$_{NP}$ Their sitting down] promises [$_{NP}$ their sitting down] to steady the canoe]]].

If this type of construction is embedded/nested in a structure of the same type, as in sentence (15), then the structure that will have to be recovered is that one in (16). The reason is that deletion occurs under structural identity—to be precise, the subject of the verb *threaten* and the subject of the clause headed by the verb *spoil* (which is a complement to the verb *threaten*) must be structurally identical, in order for deletion to apply:

(15) Their sitting down promising to steady the canoe threatens to spoil the joke.

(16) [[$_{NP}$ Their sitting down] promising [$_{NP}$ their sitting down] to steady the canoe] threatens [[$_{NP}$ Their sitting down] promising [$_{NP}$ their sitting down] to steady the canoe] to spoil the joke.

Peters (1973) and Peters and Ritchie (1973a) observe that if the parser does not have to recover a literal copy of deleted NPs, then the problem disappears. And this is precisely what Berwick and Weinberg (1984) argue in favor of. They notice that if instead of having duplicated copies of NPs, an empty category such as PRO is postulated, the output derivation grows linearly with respect to the input. The structure for sentence (15) will be instead the one in (17) below:

(17) $[_{S}\ [_{NP}$ [Their sitting down]$_{i}$ promising [PRO$_{i}$ to steady the canoe]]$_{j}$ threatens [PRO$_{j}$ to spoil the joke]]

Berwick and Weinberg observe that since (i) empty categories cannot be nested, and (ii) they correspond to displaced NPs; there can only be a fixed number of them in a sentence. Consequently, the length of the output structure does not grow exponentially. The advantage of approaching the problem this way is that efficiency in parsing is maintained while predicate argument relationships are recovered. Berwick and Weinberg observe: "an implicit representation of the tree is built without explicitly reconstructing it".[2]

The same kind of complexity argument can be offered for VPE, if the gap is detected by copying the structure of the antecedent into the gap, then ellipsis cannot be handled in a computationally efficient way. Consider sentence (18); if we represent the gap by copying the antecedent, the resulting structure is (19). But an elliptical sentence can also be embedded into a structure of the same type, (see (20)). To represent the gap in this case, all the structure in (21) will have to be copied:

(18) John said he is very happy, and Peter did too.

[2] The idea presented for ellipsis and movement operations in general, in the next section hinges on this notion of building an implicit structure without explicitly computing it. The technical notions used (movement operations, and chains) differ a little bit, but the essence of the argument is the same.

(19) John [said he is very happy], and Peter did [said he was very happy] too.

(20) John said he is very happy before Bill did, and Peter did too.

(21) John [[said he is very happy] before Bill did [said he is very happy], and Peter did [[said he is very happy] before Bill said he is very happy] too

To conclude, we have seen arguments in favor of having a structured VP (e.g. traces that need to be bound inside the elided verb phrase). However, we have also seen arguments of computational complexity against copying the antecedent's structure into the gap. So, how can the elision site be assigned some structure when the sentence is parsed, if copying is not an option? The answer I want to propose is that the elided constituent shares its structure with the antecedent.

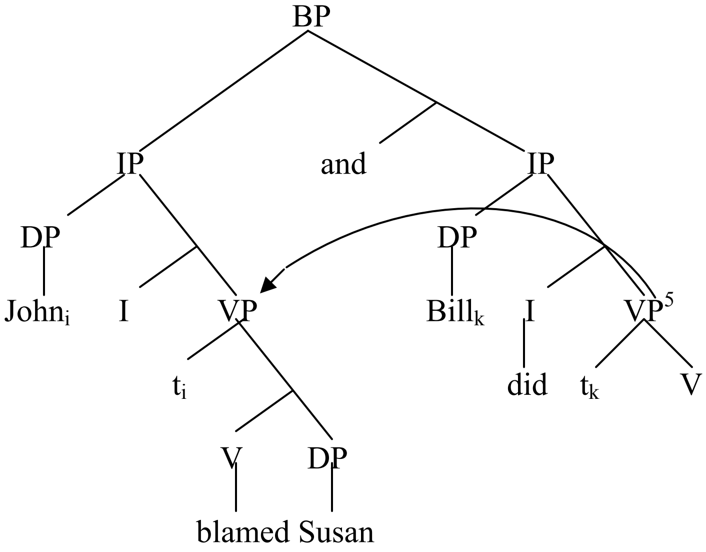
5.1.2 VPE as Sharing of Structure

I would like to propose that in the case of ellipsis, the elided VP shares the structure with the antecedent VP. The syntactic structure for the elided VP is not overtly built, but this does not mean that it is structurally empty. The elided VP is assigned a pointer to the antecedent VP: the elision site has the structure that the antecedent has;

it is structurally realized in a different position.[3] See sentence (22) and (23), which represents how the pointer assigned to the VP goes back to that structure that the antecedent VP has:[4]

(22) John blamed Susan, and Bill did too.

(23)



[3] In programming languages, if a variable is assigned a pointer to another variable, then it gets its value from the last.

[4] Frazier & Clifton (2001) propose that the human parser applies two different methods for building syntactic structure: (i) one that involves building structure step-by-step in response to the input, and (ii) Copy-α, which involves cost-free copying of the structure of the antecedent. Copying more structure doesn't necessarily cost more than copying less structure. They claim that for VPE the mechanism at work is Copy- α; it is that operation that builds the structure for the gap (Copy- α is at work when there is no ambiguity in the syntactic category to be copied).

Frazier & Clifton mention at the end of their paper that the way of implementing Copy-α is not very clear, but they suggest that one way in which it could be done is by assigning a pointer to the appropriate left bracket in the antecedent clause.

However, we have seen complexity arguments for not wanting the structure of the antecedent to be copied into the gap. The solution that I propose to that is Sharing of structure between antecedent and elision, and this is done by assigning a pointer to the elided VP that goes back to the structure of the antecedent VP. In the next sections I make explicit how this pointer assignment takes place, and what the consequences are when interpreting the elided VPs (see section 5.2.5).

[5] We will see in section 5.2.2 how this VP node is postulated for the gap. For the sake of discussion, for now, just assume that a VP can be postulated for the gap.

In order to make this relation clear I am going to look at how movement can be dealt with from a parsing perspective, and on the basis of this establish a comparison with the ellipsis cases. Consider sentence (24):

(24) [Which picture of himself$_1$] did John$_1$ like t?

In sentence (24), the DP *which picture of himself* is realized in a high position in the tree. However, it seems as if it is somehow related to the lower position *t*, at least in two ways: (i) the DP is an argument of the verb *like*, (ii) the anaphoric element *himself* is interpreted as co-refering with the noun *John*. In order not to violate Principle A of the binding theory, this anaphoric element must be related to the lower position, at least at some level of representation.

One way of accounting for those facts would be to copy the DP into the lower position, have a much bigger number of output than input symbols, and this will be unable to guarantee recursive enumerability. A second possibility is to relate the DP to two different positions; one is the grammatical position—that one in which the DP is structurally overtly realized—and the other is the logical position—that one in which the DP receives its theta-role, where it is semantically interpreted, and where its anaphoric element is bound. Thus, we have an element that is structurally overtly realized in one position of the tree, but which is related with two different positions.

The same thing is true of elliptical sentences. In sentence (22) above, repeated below as (25), the VP *blamed Susan* is only overtly realized in the first conjunct, but it is also somehow related to the second conjunct: the verb *blamed* is the predicate of the first and the second conjunct—it assigns a theta-role both to *John* and to *Bill*. In sentence (26), the verb phrase *shaved himself* is overtly realized in the matrix clause, but it is also related to the elided verb phrase position in the adverbial clause—the verb *shave* is the predicate that assigns a theta-role to the subject *Peter*. Besides, there is a reading for this sentence where the anaphor *himself* is interpreted as referring to *John* in the matrix clause, but to *Peter* in the adverbial clause. This reading can be paraphrased as: John shaved himself before Peter shaved himself. Once again, it seems that the syntactic object that is structurally overtly realized in the matrix clause is somehow related to the lower position in the tree (the elided VP), since the reflexive is interpreted there—in order to get this interpretation it needs to be bound by the DP *Peter*.[6]

(25) John blamed Susan, and Bill did too.

(26) John shaved himself before Peter did.

This concept of an element occupying two positions in the tree is expressed in linguistics by the concept of Chain. A chain is usually defined as extending in the

[6] In section 5.3, we will get back to these examples, to the idea of Chain, to how exactly anaphors are bound, and how c-command is computed for the binding of anaphors.

phrase marker over two positions <T_i,…,T_j>. The head of the chain T_i is a feature-related position—grammatical features like case, and wh-features (in the case of wh-movement), are checked here. The tail of the chain is a θ-related position—θ-roles are assigned in this position. The head and the tail must be in a c-command relation. In parsing terms, movement is justified so as to recover base-generated positions and θ-relations (in competence, movement is justified in terms of wh- and case feature checking).

The case of ellipsis is different. In example (22), the two positions that the VP *blamed Susan* is related to do not correspond to the ones defined for Chain above—the head is not uniquely a case position, nor is the tail uniquely a θ-related position. Besides, the head and the tail are not in a c-command relation.

When parsing wh-movement, a chain can be postulated as soon as a wh-element is encountered. This chain will be completed later on in the parse, when the wh-structure is related to a second position in the tree.[7]

Creating these relations between two positions in the tree for ellipsis is not as simple. First, in the case of ellipsis, it is not until we find the elided VP that we can identify the previous VP as a potential antecedent VP with two different positions in the tree. And, second, the antecedent (the "head" of the chain) can be arbitrarily far from the elision site—the antecedent can be the previous VP or there might be other phrases, even sentences, between the antecedent and the elision site.

[7] One way of doing this would be to store in a separate stack the heads of chains and then to retrieve them later on when Move is invoked, i.e., when the tail of the chain is found.

Nevertheless, both in movement and ellipsis cases there is a syntactic object that is structurally realized in one position, but which needs to be related to two or more positions in order to satisfy certain grammar constraints. A chain is built both in the case of movement and in the case of ellipsis. In the case of movement, this chain relates a constituent to two positions inside the same phrase marker. In the case of ellipsis, this chain can relate a constituent with two positions in the same phrase marker, in different phrase-markers, or even in different sentences:[8]

(27) Peter came before I did.

(28) Peter came, and I did too.

(29) A: Who came?
B: I did.

5.2 A Minimalist Account of Verbal Ellipsis

In this section, I introduce a minimalist account of verbal ellipsis. My starting point is Weinberg's (1999) human sentence processing algorithm; I extend this to account for coordination and for ellipsis sentences.

We look at different elliptical constructions—to be precise: VPE, Pseudogapping, and Gapping—and I offer an account for the different locality restrictions observed in these constructions based on (i) the presence/absence of an

[8] Chomsky (1995) defines VPE as a chain that exists at the LF level.

auxiliary, (ii) low initial attachment of coordinates, and (iii) spell-out operations which render syntactic material unavailable.

We will see that the way in which the gap is handled in VPE and Pseudogapping differs from Gapping. For the former gap resolution takes place in two steps: (i) the gap is detected by the presence of an auxiliary, and a VP is predicted on-line, and (ii) the antecedent is accessed for interpretation purposes only. In the case of Gapping, however, since there is no overt auxiliary or verb the antecedent must be accessed on-line in order to detect the presence and category of the gap.

5.2.1 Weinberg (1999): "A Minimalist Theory of Human Sentence Processing "

Weinberg (1999) assumes the minimalist program (Chomsky 1993, 1995), and applies minimalist operations—Merge, Move and Spell-Out—and minimalist principles (economy principles) to parsing. She presents a minimalist algorithm that not only accounts for some attachment preferences observed in the literature for which independent principles had been postulated—such as Minimal Attachment—but also offers a theory of reanalysis. The algorithm as presented by her follows:

(30) A derivation proceeds left to right. At each point in the derivation, merge using the fewest operations needed to check a feature on the category about to be attached. If merger is not possible, try to insert a trace bound to some element within the current command path. If

neither merger nor movement is licensed, spell-out the command path. Repeat until all terminals are incorporated into the derivation.

Weinberg (1999), following Uriagereka (1999) (see Chapter 3, section 3.1.2) , assumes a multiple spell-out (MSO) theory for performance to account for the mapping between precedence and dominance relations in parsing too (without the need for an LCA principle).[9] The base step of the LCA (in footnote 9 below), is deduced from the fact that it is the simplest mapping relation between both precedence and dominance—as we saw in our discussion of Uriagereka's work.

The induction step is not necessary if MSO applies. Weinberg assumes that Spell-out applies whenever two categories cannot be merged together. If neither merge nor move can apply, then the category being built is spelled-out, or in other words, linearized: turned into an unstructured string.[10] For this spelled-out string the only important precedence relations are those already established. Precedence does not need to be established between these elements in this string and the rest of the items in the structure. Spell-Out is an operation of the grammar, and as such it is also

[9] The LCA as proposed by Kayne (1994) derived linear precedence from dominance relations. Weinberg (1999) inverts the claim so as to make it relevant for parsing purposes:

(i) Linear Correspondence Axiom
Base step: If α precedes β then α c-commands β.
Induction step: if γ precedes β, and γ dominates α, then α precedes β.

[10] Weinberg (1999) makes clear her view of spell out as a conduit between syntax and phonology. After spell-out an unstructured string is sent to PF, but a structured unit is shipped to LF.

constrained by economy conditions. Thus, it cannot apply freely; it only applies when a chain of precedence could otherwise not be established.

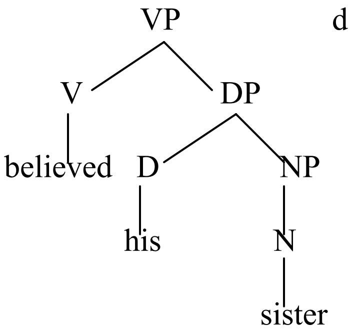
The algorithm presented by Weinberg accounts for some parsing preferences that have been discussed in the literature, and for which independent parsing principles have been postulated. It applies left-to-right and evaluates ambiguity with respect to economy conditions. Items are inserted/moved in the derivation for feature checking purposes. This explains why there is an argument-over-adjunct preference (Pritchett (1992) and Gibson (1991)), since when an item is inserted as an argument it allows features such as θ-roles to be checked, while attachment as an adjunct does not. The Minimal Attachment principle (Frazier and Rayner (1982)) is explained by the fact that all operations in minimalism are constrained by economy conditions. Thus, the preference to attach a category using the minimal possible structure follows (i.e. using the fewest steps/operations).

Consider example (31) below, discussed by Weinberg (1999). The verb *believe* subcategorizes for an NP (as in (31)), or an IP (as in (31)). When parsing these two sentences, at the point where the determiner is encountered, the grammar provides two options: (i) attach the DP as the object of the verb *believe* (in (31))—by which the DP will be assigned case and θ-theory, or (ii) attach it as the subject of the embedded IP (in (31))—in this case no case or θ-checking will take place at this point, since the θ-assigning head (the verb of the embedded clause) has not been encountered yet:

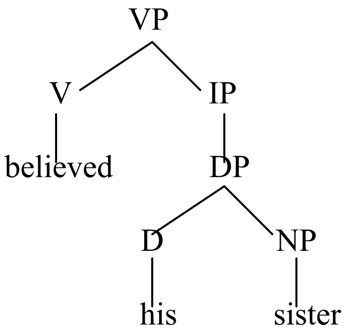
(31) a. The man believed [$_{DP}$ his sister].

b. The man believed [$_{IP}$ his sister to be a genius].

c.



d.



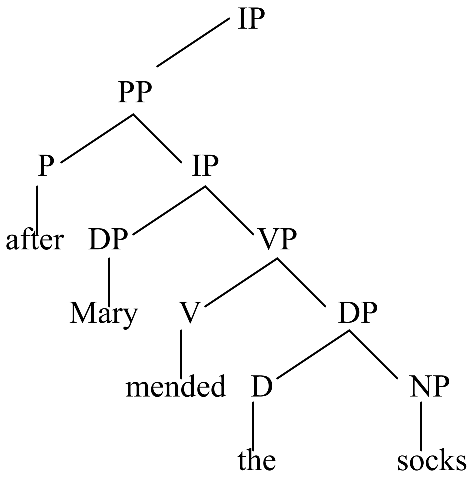
Thus, structure (31) is preferred for both sentence (31) and (31), since feature checking is optimized. It is also the most economical structure (fewest nodes). In the case of sentence (31), when the embedded verb is encountered, the structure assigned to the sentence will have to be reanalyzed into the structure in (31). But, this is possible since spell-out has not occurred and both the verb and the object DP are available for reanalysis.

Weinberg's algorithm also offers a theory of reanalysis. Reanalysis to a different reading remains possible within a domain where spell-out has not applied (as in (31)); the preferred reading can be transformed into the dispreferred reading. This is not possible, however, if spell-out has applied. Then, extraction of material from, or insertion of material into, the spelled-out unit is impossible. The reason for this is that once a structure is spelled-out it is frozen, and cannot be affected by operations such as merge or move. The result of spelling-out a constituent is an unstructured string, that is, a string with no internal phrase structure. Example (32), also discussed by Weinberg, illustrates this point:

(32) After Mary mended [$_{NP}$ the socks] fell off the table.

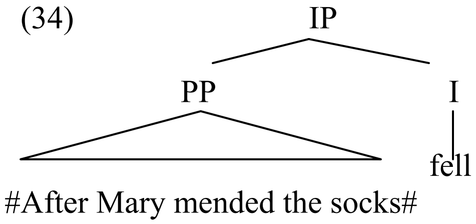
For this sentence there are two possibilities: (i) attachment of the DP *the socks* as the object of the verb *mend*, or (ii) attachment as the subject of the matrix clause. The DP *the socks* is attached as the object of *mend* (in (33) below), for the same reasons as in the above example—i.e., checking of features in the DP by the verb. The second option, attachment as the matrix subject, does not allow any checking of features, since the head of the IP is not part of the structure yet.

(33)



Since the next word *fell* cannot be merged with anything in the preceding clause, the relationship between precedence and c-command is broken. Therefore, in order to attach the next word, the verb *fell*, the current structure needs to be spelled-out. Otherwise, it will be impossible to merge this item and maintain the precedence-command relation. The adverbial phrase, therefore, is spelled-out and the verb is attached. The resulting structure is the one in (34) below:

(34)



Reanalysis in this case is not possible, because the domain where the DP is attached has been spelled-out, and consequently, it is not possible to retrieve it in order to attach it as the matrix subject.

In a nutshell, Weinberg (1999) presents a parsing theory based on minimalist operations that accounts for certain attachment preferences, and which also offers a theory of reanalysis which piggybacks on whether the syntactic material to be reanalyzed has been spelled-out or not.

5.2.2 The Auxiliary Effect

There exists a difference between VPE and Pseudogapping elliptical constructions on the one hand, and gapping on the other. In the case of VPE and Pseudogapping, there is an auxiliary overtly realized (see (35) and (36)). On the contrary, in gapping sentences there is no auxiliary present (see (37)):

(35) Mary is very hungry, and I am too.

(36) Peter gave his corrections to Susan, and John did to Bill.

(37) These students ate bagels, and the visitors pizza.

This difference is crucial for detecting and resolving the gap. In the case of VPE and Pseudogapping, since there is an auxiliary, an IP can be built and a VP predicted (functional categories like "I" select lexical categories like "V"): the auxiliary is recognized on the basis of the input string (bottom-up,) an IP is built, and a top-down prediction of a VP can be made. All this is done by using local context, i.e. the information provided by the auxiliary. There is no need to access the antecedent to detect the gap and assign structure to it. However, in the case of gapping, there is no overt auxiliary or verb from which to build an IP, and the antecedent needs to be accessed in order to detect the possibility of a gap. The antecedent is needed to postulate a node for the gap. This Tense effect was already noticed by Fodor (1985), and discussed by Berwick and Weinberg (1985).

In VPE and Pseudogapping sentences, the VP that is predicted is assigned a pointer to the antecedent VP and it shares the structure with the latter. The antecedent structure is accessed only for interpretation. It is accessed not on-line to build the structure of the gap. For gapping, the antecedent is accessed on-line to assign structure to the gap.

Frazier and Clifton (2001) report what they call "missing complexity effects" in VPE sentences. In a self-paced reading experiment, they did not find any difference in the reading times of those sentences in (38), even though the structure of the antecedent in (38) is more complex than in (38):

(38) a. Sarah left her boyfriend last May. Tina did too.
b. Sarah got the courage to leave her boyfriend last May. Tina did too.

This contrasts with complexity effects found by Carlson (2002) for gapping sentences. This difference between VPE and gapping sentences supports the distinction that I have proposed above for VPE and gapping. In the case of VPE, it seems that the antecedent is not accessed for gap detection; otherwise, if the structure of the antecedent is computed for the gap, then there should be differences in the reading times.

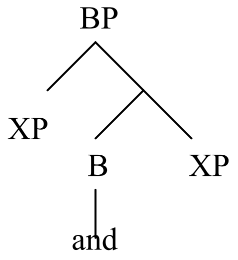
5.2.3 Low Initial Attachment of Coordinates

Weinberg (1999) evaluates ambiguity of attachment with respect to economy: the most economical structure is preferred, i.e. that one that involves fewest nodes or operations. I translate this economy preference into initial low attachment for coordinates.

Coordinators are initially attached low, and this decision is revised into high attachment if later incoming material obliges us to reanalyze—we will see how reanalysis is carried out in detail when I discuss some examples in the next section.

I assume that coordinates head Boolean Phrases (BPs) as proposed by Munn (1987)—coordinate sentences have the structure in (39) below. As proposed in Murguía (2000), coordinates are spelled out in different CUs in order to preserve the precedence-command relationship among terminal elements.

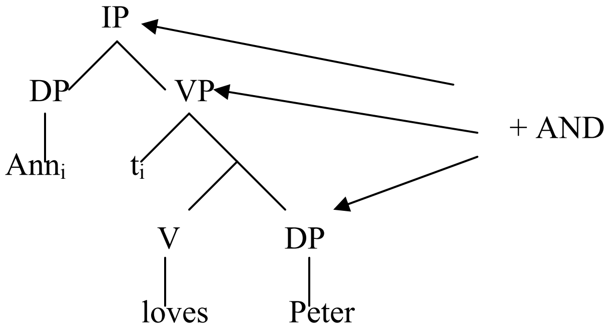
(39)



Now, let us consider why low attachment is more economical than high attachment. For a sentence like (40), at the point where the coordinator is encountered the structure computed so far is that one in (41), where the structure of what is going to be the first conjunct is already built. At this point, the next input item to be attached is the coordinator *and*. How is the coordinator attached? There are three possible attachment sites, marked with arrows—the three possible attachment sites are the (i) IP, (ii) VP, and (iii) DP nodes:

(40) Ann loves Peter, and Mary does too.

(41)



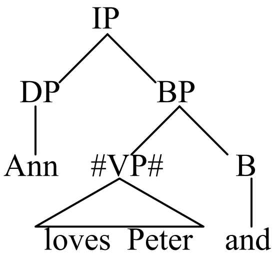
How does the parser choose among these possibilities? Recall that I am assuming a minimalist grammar, and the most important principle in minimalism is the principle of economy—derivations must be as economical as possible (fewest number of steps/operations and fewest number of nodes). This economy principle is what is going to guide the parser in choosing among the three alternatives. Let us consider these in turn. Using the strategy of actually considering all the possible alternatives will be a problem for efficiency in parsing.[11] Therefore, economy is enforced in a serial way. Low attachment is the initial choice the parser takes, because it is more economical. The goal of the discussion that follows is to illustrate why it is the case that low attachment of coordinates is more economical. This then justifies the assumption that it becomes the automatic first option without the need for global comparison.

We will start considering alternative (ii): attachment to the VP. In order to attach the coordinator to the VP node and preserve the command-precedence relationship, the intervening material must be spelled-out, so that this position

[11] If all the different attachment items were considered on-line, then the algorithm will be n^2 proportional to attachment sites, which does not respect efficiency.

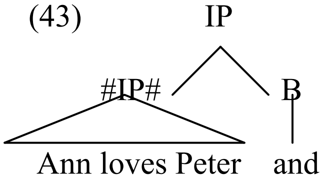
becomes available. Since spell-out occurs in a phrase-by-phrase manner, the DP will be spelled-out first, followed by the VP. After these two spell-out operations, the coordinator can be merged to the structure. This is what the structure will look like after all these operations:

(42)



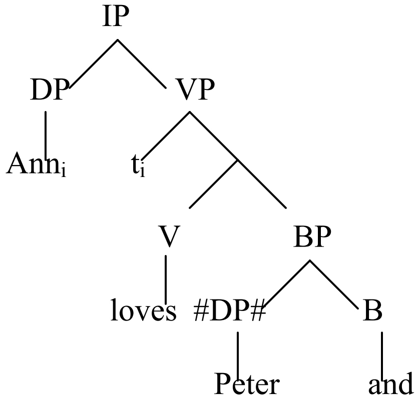
Alternative (i), attachment to the IP node, will include the same steps as attachment to the VP plus one more spell-out operation—spell-out of the IP. The structure after attaching the coordinator will be that in (43):

(43)



Turning now to the third possibility: attachment to the DP. If the coordinator is attached to the DP then only one spell-out operation is necessary: spelling-out of the DP as in (44) below. Thus, this third possibility is the most economical one (it involves fewest steps/operations), and the one that the parser chooses—decisions are taken locally, this algorithm is not a global one, and at this point in the derivation attachment to the DP is the best option in terms of economy:

(44)



Thus, according to economy, it looks like low attachment should be preferred for coordinates too.[12] This tendency for attaching low has already been observed in the parsing literature in other contexts different from coordination (Minimal Attachment (Frazier and Rayner (1982))) and explained in terms of a minimalist parsing algorithm that favors feature checking, and respects economy ((Weinberg (1999)) above).

5.2.4 Locality Effects Revisited

It is a well-known fact in the literature that elliptical constructions are subject to different locality restrictions (e.g. Chao (1987), Fodor (1985), and Berwick &

[12] We will see how this low attachment preference in coordinates has an effect on the preferred readings in cases of gaping in the next section.

Weinberg (1985)). VPE and Pseudogapping are not constrained by any locality restrictions, while gapping is:

(45) a. Ann loves Peter, and Mary does too.

b. Ann loves Peter, and Susan thinks Mary does too.

(46) a. I gave money to Susan, and Peter did to Beth.

b. I gave money to Susan, and you heard that Peter did to Beth.

(47) a. John saw Carmen, and Tom Othello.

b. *John saw Carmen, and Bill thinks Tom Othello.

Both for sentences (45) and (46) (examples of VPE and pseudogapping, respectively) the antecedent verb phrase, and the elided constituent can be separated by intervening material—the gap can be embedded (as in the "b" examples), and still result in a grammatical sentence. However, in the case of gapping (sentence (47)), if the antecedent and gap are not local (if the elided constituent is embedded as in (47)) then the sentence is ungrammatical.

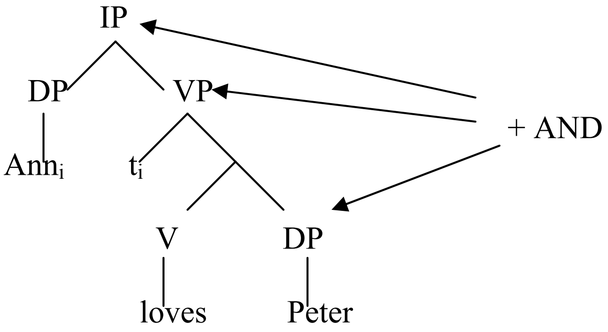
In this section, I propose an analysis for the presence/absence of locality effects in ellipsis which is based on (i) the presence/absence of the auxiliary, and (ii) the availability of left context (i.e. of the antecedent), which in turn is a result of low initial attachment of coordinates and of spell-out operations that render syntactic structure unavailable.

5.2.4.1 VPE

Let us start with the VPE example (48). Through subsequent Merge and Move operations, the first conjunct structure is built. The next input item to be attached is the coordinator *and* and the three possible attachment sites are those in (49):

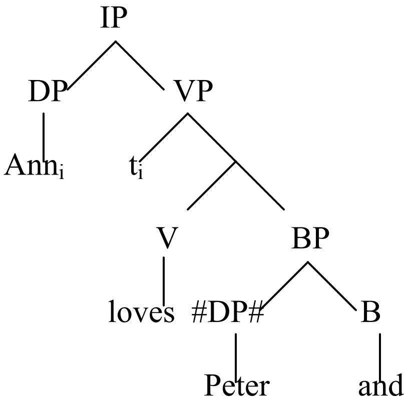
(48) Ann loves Peter and Mary does too.

(49)



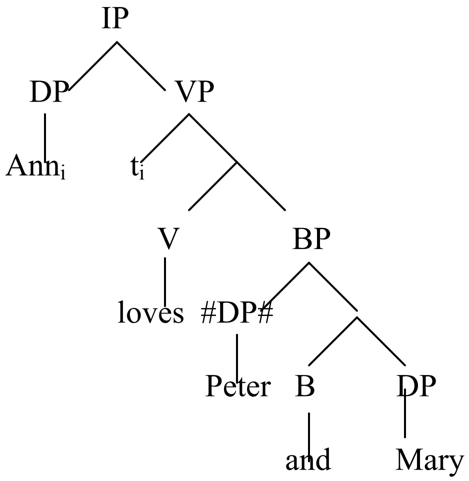
As we saw in the previous section, the parser chooses low attachment (i.e. attachment to the DP), since this is the most economical option; the one that involves fewest steps.

(50)



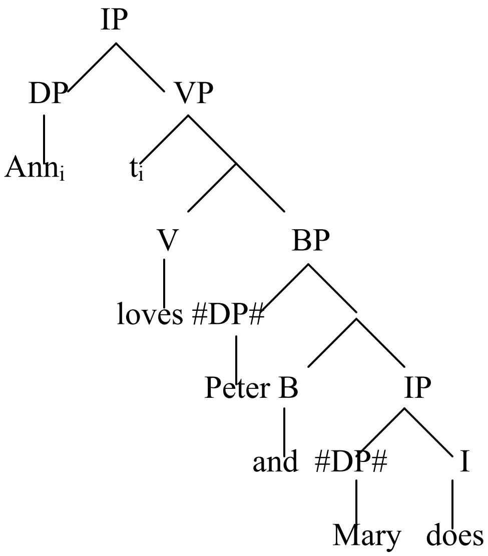
Once the coordinator has been merged to the structure, the next input item to be attached is the DP *Mary*, which will be attached as follows:

(51)



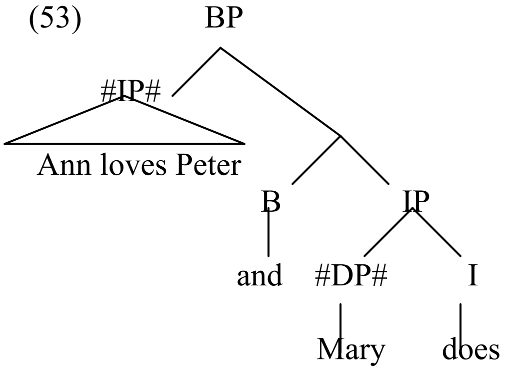
There is a condition on coordination that must be respected; the coordinator must con join two identical categories. (Coordination of Likes: Williams (1981)). If two different categories are coordinated, then this constraint is violated and the sentence is ungrammatical. In (51) above, two DPs are coordinated, so the condition on coordination of likes is respected. The next input item to be attached is the auxiliary *does*. As before, the most economical option is to attach low:

(52)



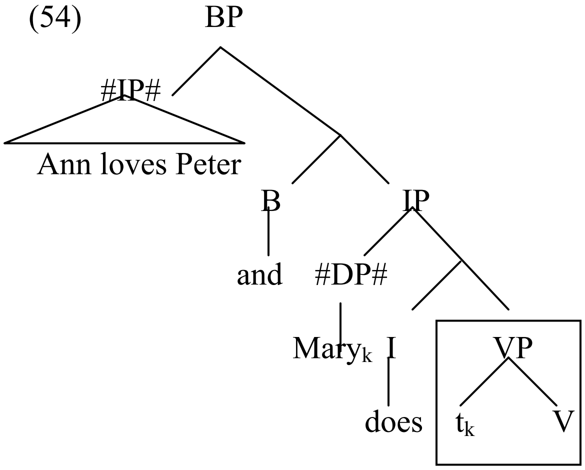
However, one more node (IP) had to be postulated (the DP *Mary* that was coordinated with the DP *Peter* has been transformed into an IP). Now *and* is coordinating a DP *Peter* and an IP *Mary does*, which violates the condition on coordination. At this point reanalysis is necessary. Low attachment is reanalyzed as high attachment, in other words, coordination of objects is reanalyzed as coordination of IPs. To respect the condition on coordination there is only one possibility now, and that is attachment to the IP, as in (53) below:

(53)



To attach high as in (53) above, the whole IP needs to be spelled-out so as to preserve the precedence-command relationship among terminals. At this point, where we have an IP in the structure, we can predict a VP (since functional categories select lexical categories). We do not need to look back to the antecedent to keep on building structure. We can also relate the subject in the specifier of IP to its base position—the specifier of VP—and build all the structure in (54) below:

(54)



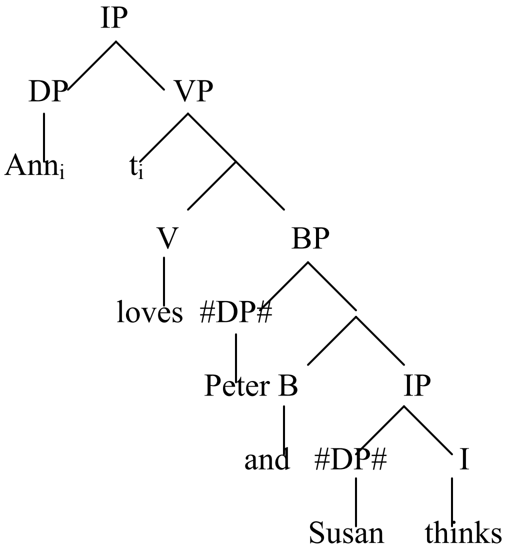
The complete internal structure of this VP, however, as well as the lexical content of the V-category and its complement are not fully specified. The rest of the VP structure and the lexical content are recovered from the antecedent, by following the pointer that the elided VP is assigned. So, the antecedent is retrieved for interpretation purposes at LF, but not on-line when the gap is encountered.

Consider now the second example of VPE I mentioned above—where the antecedent and the elided VP are separated by intervening material—repeated here for the reader's convenience:

(55) Ann loves Peter, and Susan thinks Mary does too.

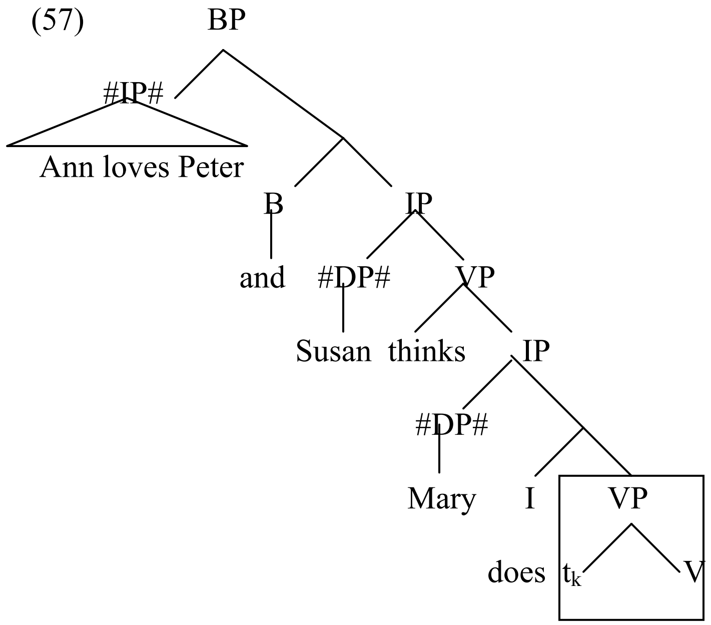
This sentence will be parsed in the same way as the previous example—the coordinator will be attached low, since this is the most economical option. Reanalysis from low into high attachment here, however, will be triggered by the attachment of the intervening clause *I think* in (56) below:

(56)



At this point, the condition on coordination is not satisfied (a DP and an IP are coordinated) and reanalysis is necessary. The coordinator is attached to the higher IP. In order to attach high, the antecedent clause needs to be spelled-out, consequently, the antecedent VP will not be accessible—the antecedent cannot be retrieved to assign structure to the gap. Nevertheless, this is not a problem for VPE examples, since a VP can always be predicted from the IP, without resorting to the antecedent's help:

(57)



5.2.4.2 Pseudogapping

Consider now the pseudoggapping (or subdeletion) examples mentioned above, which are repeated below:

(58) a. I gave money to Susan, and Peter did to Beth.

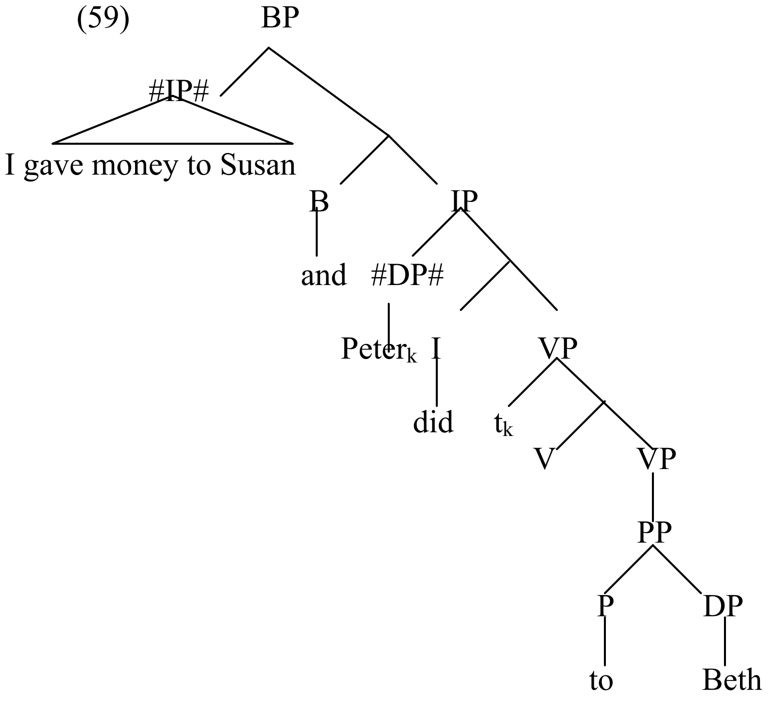
b. I gave money to Susan, and you heard that Peter did to Beth.

Pseudogapping sentences, like VPE, are grammatical whether the antecedent and the elided clause are local or not. In (58) for example there is intervening material between both clauses, but the sentence is still grammatical. An auxiliary is always present, as in the case of VPE, so an IP is built and a VP can be predicted without the need to access the antecedent.

One difference between VPE and pseudogapping examples is that one of the verb arguments/adjuncts in the elided conjunct is overtly realized only in the latter. In example (58) above, the indirect object *to Beth* has not been elided. How is this overtly realized argument attached, when the verb phrase is elided? Since a VP is predicted for all pseudogapping cases the argument is attached as part of that predicted VP.

Consider the parse for sentence (58). The argument will run in the same way as for the VPE examples above—attachment of the coordinator starts low and is reanalyzed to IP attachment when the auxiliary (in (58)) and the clause *I heard* (in (58)) are attached. Once the second conjunct is reanalyzed as an IP, a VP is predicted. Finally, the non-elided argument is merged to the structure:

(59)



The sentence in (58) is parsed in the same way, with the difference that reanalysis in this case is triggered by the intervening material (as in (55) above). But, since an auxiliary is present in the elided constituent an IP is built, and a VP predicted—to which the overtly realized argument is attached. Because of this possibility to predict a VP, the non-availability of the antecedent (it has been spelled-out, so it is not available) does not pose a problem neither for the resolution of the gap nor for the attachment of the argument *to Beth*.

5.2.4.3 Gapping

The gapping examples differ from VPE and pseudogapping by showing locality effects. When the antecedent and the elided clause are separated by intervening material (as in (60) below), then the sentence is ungrammatical:

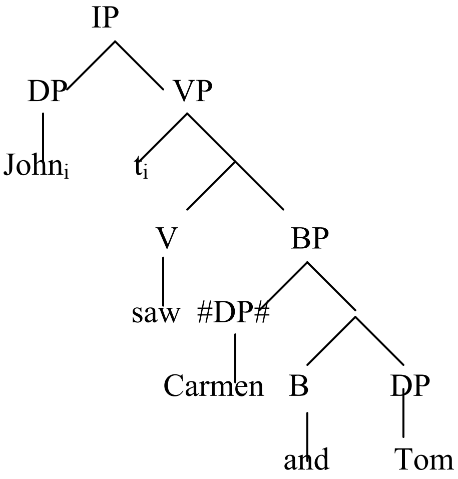
(60) a. John saw Carmen, and Tom Othello.

b. *John saw Carmen, and Bill thinks Tom Othello.

With the minimalist parsing algorithm that I have assumed, particularly with a theory of MSO where material is rendered inaccessible for further computation after being spelled-out, I can account for why these locality effects are observed in gapping.

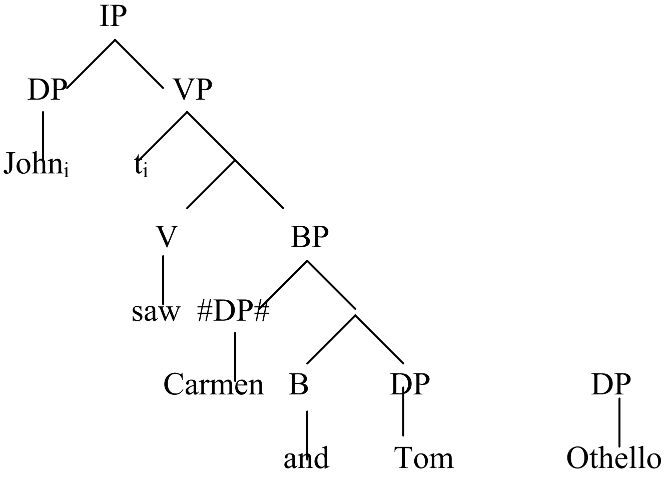
Consider sentence (60). The first conjunct is parsed and the coordinator *and* once more is attached low for economy reasons. The next item attached is the DP *Tom*, which is attached low, as a coordinated object. The structure at this point looks like:

(61)



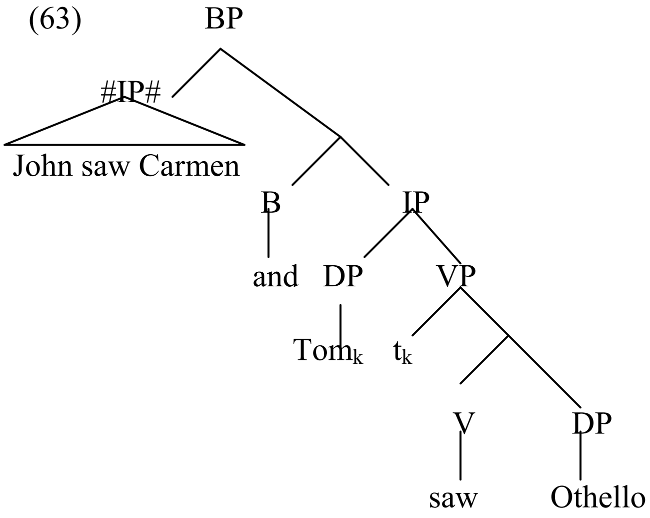
Now, the next word to be attached is the DP *Othello*, but there is no way in which it can be attached to the structure. In this case, we do not have an auxiliary or verb in the current clause, as in the VPE or Pseudogapping examples, that will help us predict a VP. The parser needs to go back to the antecedent clause and use the information about the predicate in that antecedent clause to relate the two arguments *Tom* and *Othello*:

(62)



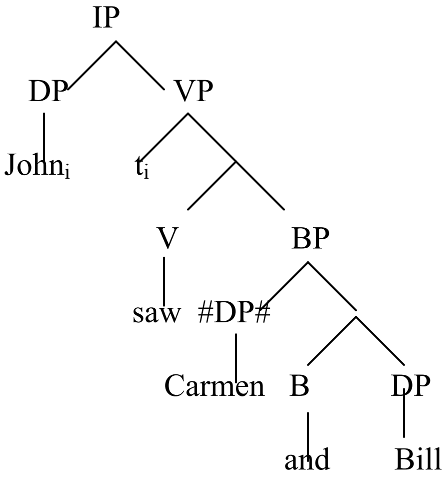
The parser reanalyzes the structure by looking for an antecedent in the c-command path. The antecedent is still available because of the initial mistake of attaching low triggered by economy. The coordination attachment is reanalyzed and the resulting structure is that one in (63) below:

(63)



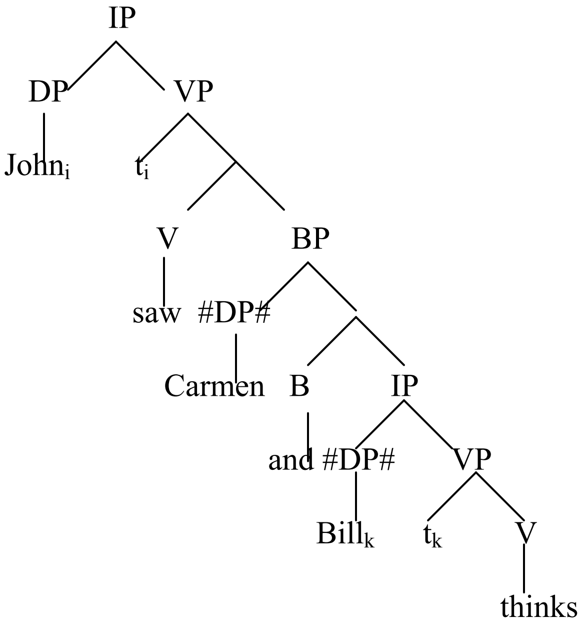
However, in the case of (60) the verb gap cannot be reconstructed, because by the time the parser gets to the gap the antecedent has already been spelled-out. Let us see this in some more detail. The coordinator and the DP *Bill* are attached low (for the same reasons I have claimed for the previous examples), as in (64):

(64)



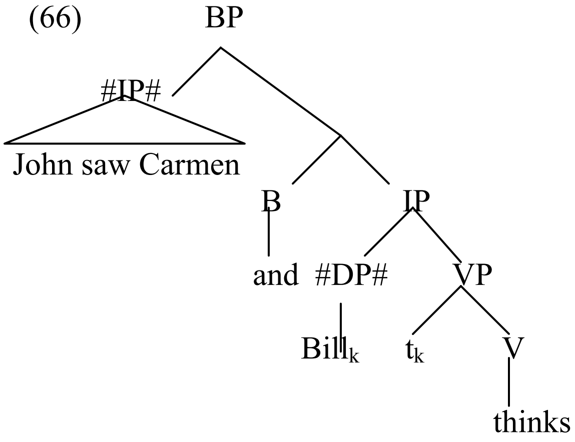
Reanalysis for sentence (60) is triggered when the verb *think* is merged to the already existing structure, since the resulting structure violates the condition on identity of categories for coordination:

(65)



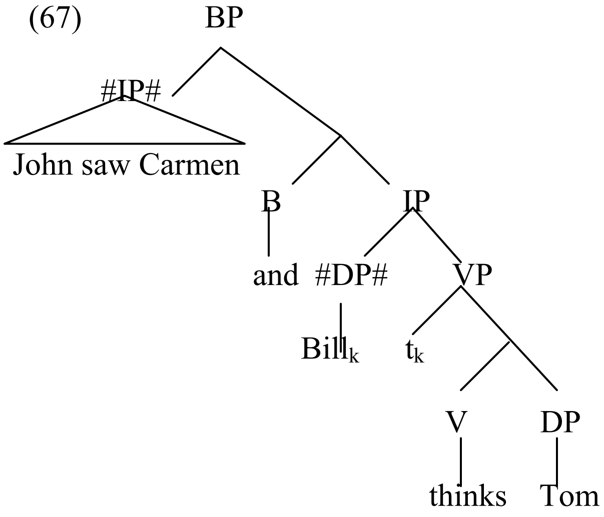
This structure is reanalyzed as follows. We have an IP so the only possibility is to attach as a coordinated IP. To do so, the first conjunct (i.e. the antecedent) is spelled-out, in order to preserve the command-precedence relations:

(66)



The parse will follow by attaching the DP *Tom* as in (67) below, but then the next word *Othello* cannot be attached to the existing structure in any way. An auxiliary is not present in the elided clause, as in the VPE or pseudogapping cases, so a VP cannot be predicted and the gap cannot be interpreted. Because the antecedent has already been spelled-out, it cannot be accessed to license the gap, resulting in an unacceptable sentence:

(67)



Thus, we have seen how the locality effects that gapping cases display can be explained in terms of MSO, and of the minimalist algorithm that I am assuming. In gapping the antecedent and the elided clause must be local; otherwise, the gap cannot be licensed. The gap depends on the antecedent to be reconstructed.

5.2.5 Low Attachment of Coordination and Case Marking

In the previous section, I have proposed that coordinates are initially attached low, and I have accounted for the different locality restrictions in ellipsis phenomena based on this and other factors. A question that could be raised now for this proposal of initial low attachment of coordination is whether it makes the right predictions beyond the cases discussed here.

It has been noted that case marking has a bearing on processing; it affects attachment decisions made by the parser. In sentences where there are different possible attachment sites, the ambiguity effect seems to disappear with case marking. Consider the examples below:

(68) John knows Mary likes movies.

(69) John knows she likes movies.

In sentence (68), there are two attachment options for the DP *Mary*; it can either be attached as the object of the verb *knows* in the main clause, or as the subject of the embedded clause headed by the verb *likes*. This ambiguity disappears in example (69), where nominative case marking on the pronoun signals the attachment of *she* as the subject of the embedded clause to be the right option.

The question now is what happens with coordination cases when there is case marking information. In sentences like (70), (71) and (72) the coordinator is followed by a pronoun which is marked with nominative. Does nominative case marking affect parsing decisions here as well?

(70) Ann loves Peter, and she does too.

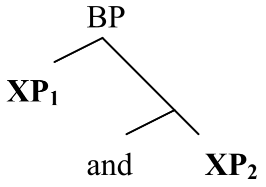
(71) I gave money to Susan, and she did to Peter.

(72) John saw Carmen, and he Othello.

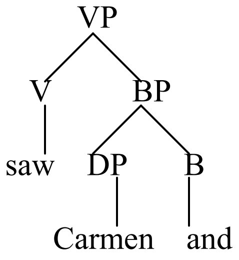
Let us assume for the sake of discussion that case marking indeed also affects parsing decisions in the case of coordination. If that is the case, the coordinator will still be attached low (I am not assuming lookahead for the parser, so at the point the coordinator needs to be attached, the parser does not have any information about the next input item). However, the nominative case marked pronouns will trigger reanalysis of the coordination. Thus, low attachment is reanalyzed as high attachment (to the IP); this reanalysis forces the antecedent to be spelled-out in a phrase-by-phrase manner.

I would like to suggest that, even though case marking influences parsing decisions, the case with coordination might be different. In coordination there is another factor that comes into play and that can compete with case marking effects: coordinators subcategorize for two or more elements (Goodall 1987), they link/coordinate two phrases (as in (73) below). So, in any of the examples above, after the coordinator is attached low, this requirement of linking two elements is partly satisfied. In sentence (70), for example, after the coordinator is attached low, one of the elements that it links is the DP *Carmen*:

(73)

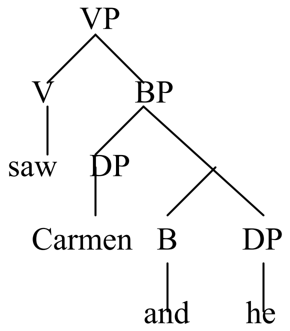


(74)



However, there is still a second element that needs to be attached as part of the coordination, in order to satisfy the Boolean Phrase constraint. This requirement can immediately be satisfied by the pronoun *he*—without the need of extra work for the parser. Thus, with coordination, the need to supply the coordinator with two phrases (two DPs in the case under discussion) and to do that in an economical way might play a more important role than case marking and influence the parser decisions with respect to attachment. The decision of attaching low, as the second element of the Boolean phrase, is a structural decision and it might not be affected by case marking.

(75)



If instead of attaching the pronoun as the second coordinated phrase, nominative case marking prevents it, and suggests projecting an IP instead, then the Boolean Phrase requirement could not be satisfied immediately. Reanalysis would force us to spell-out the antecedent in a phrase-by-phrase manner, and only then, after coordination is reanalyzed as IP coordination, can the Boolean Phrase be complete.

In a nutshell, what I am proposing here is that there is a competition between the structural requirement of the Boolean Phrase on the one hand, and nominative case marking on the other. The requirement (i) to supply the BP with the two phrases that it needs, and (ii) to do it in an economical way is stronger than the influence of case marking in coordinates, and it tells the parser to attach low, ignoring case.

5.2.6 Interpreting the Elided VPs

In this section, I discuss how the interpretation of the elided VP is carried out. I have argued that in VPE and Pseudogapping cases the gap is interpreted in two steps (i) on-line prediction of a VP, and (ii) interpretation of that predicted VP by accessing the antecedent.

For the interpretation process I assume Lappin and McCord's (1990) algorithm (see Chapter 4, section 4.1.2.2), which goes back to the antecedent and for each argument that the antecedent's predicate has it checks whether that argument is realized in the elision site. If not, then the gap inherits that argument from the antecedent. Before including the algorithm and discussing some examples let us introduce an important concept: lexical structures.

5.2.6.1 Lexical Structures instead of SlotFrames

We have seen when we discussed Lappin and McCord (1990) how VP anaphora resolution is described as the relation between the head verb and selected arguments and adjuncts of a structured empty or partially empty VP, and the head and corresponding adjuncts of an antecedent VP.

The algorithm they propose makes use of the so-called “SlotFrames”. In Minimalism, we have a data structure similar to a SlotFrame associated with every head word in the lexicon: subcategorization information, where every verb is associated with a number of obligatory and optional arguments of a certain category type. This subcategorization information is not copied into the head verb node in the tree, as it is in Slot Grammars, but it is recoverable from the tree structure.[13] Assuming a relativized version of The Uniformity of Theta Assignment Hypothesis (UTAH) we can recover the argument structure of a verb from the syntactic tree.[14] The definition of RUTAH follows:

(76) Relativized UTAH (RUTAH) (Baker 1997)
Identical thematic relationships between items are represented by identical relative hierarchical relationships between items at the level of D-structure.

[13] In Slot Grammars each head node in the tree is assigned not only a variable name (like the rest of the nodes, which allows to refer back to every node in the tree), but also the SlotFrame with which it is associated—that is, a list of all the arguments and adjuncts marker variables dependent on that head.

[14] I am assuming Larson (1988) analysis of verb phrases and dative alternation, where the double object construction in (ii)a is derived from the double complement construction in (ii)b:

(ii) a. John gave a book to Mary.
b. John gave Mary a book.

Thus, by looking at the tree structure of the antecedent we can extract the lexical structure of that predicate, and use it to interpret the elided sentence. It is necessary to look at the tree itself, rather than at the subcategorization for that verb in the lexicon, since there are optional arguments that may not be realized in the antecedent, and therefore, cannot be part of the elided structure. I am going to assume that when parsing a sentence besides building a syntactic tree, there is also an analog representation being built, that is, the lexical structure of a predicate.

5.2.6.2 Some Examples

To interpret the elided VP, the antecedent's lexical structure is retrieved. The steps that are followed to do so are defined in (77) (the algorithm below is adapted from Lappin & McCord (1990)):

```
(77)  For each argument $Argument_i$ in the lexical structure of A (the
      antecedent) filled by a phrase C
         If there is a phrase C' in VP' (the elided VP) which is of the
         appropriate type for filling $Argument_i$ , then fill it in the lexical
         structure of VP' with C'
         Else
            Fill $Argument_i$ in VP' with the address of C
```

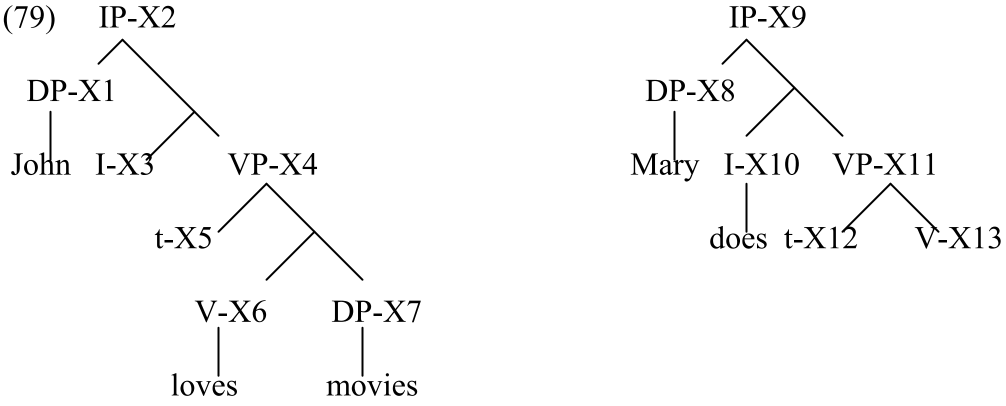
```
For each adjunct Adjunct of A
    If there is no adjunct of the same type as Adjunct in VP’, then
    include Adjunct as a new adjunct of VP’
```

Let us consider the algorithm at work with some examples. I will start by discussing an example of VPE. Observe sentence (78) below:

(78) John loves movies and Mary does too.

The structure for the first conjunct is built first. When building the structure for this, each node in the tree receives an address/marker variable (e.g. X1, X2,…Xn), so that it is possible to distinguish them and also to refer back to each of them at later stages in the parse if necessary. Once the first conjunct is parsed, the second conjunct structure is built. Recall that first the coordinator and the second conjunct subject are attached low, and that this low attachment needs to be revised as high attachment when the auxiliary is incorporated (see discussion in section 5.2.4). After reanalyzing, a VP is predicted from the IP. All the structure in (79) can be built for the elided verb phrase without resorting to the antecedent. This is what the structure for the two conjuncts will look like:

(79)



Now the antecedent still needs to be accessed to finish interpreting the gap. This interpretation process occurs in the following way: (i) an antecedent is searched for, (ii) the elided verb phrase is assigned a pointer to the antecedent verb phrase (so VP-X11 is assigned a pointer to VP-X4), (iii) the antecedent's lexical structure is retrieved and used to interpret the elided verb phrase. This last step is identical to the resolution algorithm that Lappin and McCord (1990) propose; the difference is that our algorithm is able to predict and build some structure before reaching this point.

Let us describe briefly step (iii) proposed above. The retrieved antecedent lexical structure is that one in (80), and it is used for the interpretation of the elided VP by following the steps of the algorithm described in (77). After this process has been completed, the lexical structure assigned to the elided VP is the one that follows in (81):

(80) (love (args Agent DP-X1:John
Theme DP-X7:movies))

(81) (love (args Agent DP-X8:Mary
Theme @DP-X7))

The elided VP Agent argument *Mary* is structurally realized in the elided clause, the object argument, however, is assigned a pointer to the object argument of the antecedent (represented as @DP-X7). The object is not copied into the gap, but rather inherited: its position in the lexical structure is filled with the address of this element in the antecedent verb phrase structure. Thus, the elided VP receives a complete interpretation, but no structure is computed for the non-overt elements. The non-overt elements are assigned to the elision site with the use of pointers that go back to the antecedent.

This two-fold process in which some structure is computed on-line (i.e. the predicted VP), and then the antecedent is used to interpret the partially built VP will allow us to parse elliptical sentences in a efficient way, since only as much structure as necessary is computed for the gap (i.e. enough structure to bind traces or to attach overt elision remnants). So if it is the case that a trace does not need to be bound inside the elision site nor an overt argument need to be attached to the elided VP, then the structure that is built is a VP with the subject related to its base-generated position. If, on the contrary, there is a trace or an overt argument, then enough structure so as to accommodate those is built.

We now look at one example of trace binding and one of an overt argument. Consider sentence (82) first, where there is a trace that needs to be bound inside the elided VP. As in the case of the other VPE examples we have discussed, a VP will be predicted from the IP, and both the subject and the moved wh-element will be related to their base positions:

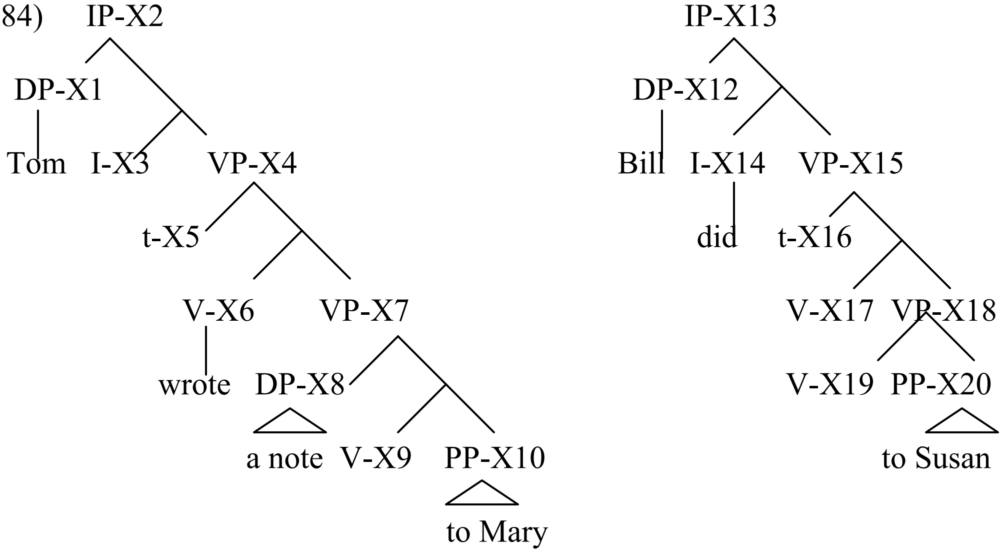
(82) I know who Mary$_i$ [$_{VP}$ saw t$_i$], and who$_m$ Amy$_k$ did [$_{VP}$ t$_k$ V t$_m$] too

In this case, when retrieving the antecedent's lexical structure, no argument will be inherited from the antecedent, since both the subject and the object have a correlate in the elision site, realized as traces.

Consider sentence (83) now, an example of pseudogapping. In this case as well, our algorithm will predict a VP from the IP. For this sentence, more structure is going to be computed online, since there is an overt argument *to Susan* that needs to be attached:

(83) Tom wrote a note to Mary, and Bill did to Susan.

(84)



The lexical structure of the antecedent will be retrieved as in (85) and used to finish interpreting the elided verb phrase as in (86):

(85) (write (args Agent DP-X1:Tom
Theme DP-X8: a note
Patient PP-X10: to Mary))

(86) (write (args Agent DP-X12:Bill
Theme @DP-X8
Patient PP-X20: to Susan))

The lexical structure of the elided verb phrase in (86) shows how the Agent and the Patient are the DP *Bill* and the PP *to Susan* respectively, which are overtly realized in the clause. It also shows that the Theme argument is assigned a pointer to the theme argument of the antecedent.

Consider sentence (87), an example of ACD, below. In this sentence, a trace needs to be bound inside the elided verb phrase. The structure that is built for the elided verb phrase is the following one:

(87) Mary read everything which$_n$ Peter$_i$ did [$_{VP}$ t$_i$ V t$_n$].

The parser will recognize at the point where *which* is attached that it has to postulate a trace (later on in the parse). That trace is inserted after the auxiliary is attached, and the VP is predicted; it will be attached as part of the elided VP. This is very important, since it is going to avoid the interpretive regress that will obtain if the whole antecedent VP lexical structure is used to interpret the gap. The lexical structure of the antecedent will be then retrieved and used to interpret the elided verb phrase as in (88); the Theme of the elided verb phrase will be recognized to be a trace—the antecedent theme argument does not need to be assigned to the gap. This trace is already inserted by the time that the antecedent's lexical structure is accessed:

```
(88)    (read   (args  Agent DP-X1: Mary
                       Theme DP-X5: everything which_i
                                  (read   (args   Agent DP-X10:Peter
                                                  Theme: trace_i))))
```

5.2.7 Gapping and Pseudogapping with Ditransitive Verbs

We now look at an interesting contrast that exists between gapping and pseudogapping constructions with ditransitive verbs. The theory that I have been assuming, a minimalist theory that prefers economical structures to less economical ones—in other words, that starts attaching low—predicts this difference.

5.2.7.1 Gapping Constructions with Ditransitive Verbs.

There seems to be a preference for English speakers to interpret the sentences below as involving the coordination of two objects—in other words, a preference to assign the non-gapped reading. Consider the sentences below and their possible readings; in all the examples the preferred reading is the one in "a", rather than the gapped reading in "b":[15]

(89) John sent a letter to Mary, and Peter a message.

(90) a. John sent a letter to Mary, and John sent Peter a message.

b. John sent a letter to Mary, and Peter sent a message to Mary.

[15] We do not have any experimental evidence for this claim, but Carlson (2002) studies the processing of gapped structures, like the ones below, both in a written and in an auditory questionnaire. She reports a preference for the non-gapped reading, which she explains in terms of a preference for minimal structure:

(iii) John visited Marjorie during the vacation, and Sarah during the week.
(iv) Sam gave a note to Sarah, and John to Sue.

(91) John sent a letter to Mary, and Peter to Susan.

(92) a. John sent a letter to Mary, and John sent Peter to Susan.

b. John sent a letter to Mary, and Peter sent a message to Susan.

(93) Mary sent John some books, and Susan Peter.

(94) a. Mary sent John some books, and Mary sent Susan Peter.

b. Mary sent John some books, and Susan sent Peter some books.

(95) Mary sent John some books, and Susan some documents.

(96) a. Mary sent John some books, and Mary sent Susan some documents.

b. Mary sent John some books, and Susan sent John Peter.

We can see from the examples above that the preferred reading is that one in which attachment of the second conjunct is low: it is interpreted as a coordinated object. Low attachment is justified in terms of economy, and this is a prediction that our algorithm will make for these cases.

Consider now the pseudogapping examples below (the material between brackets represents those elements that have been elided. All the sentences below are grammatical, except for (100)—a double object construction with the first object elided:

(97) Peter sent a Christmas present to his girlfriend, and John did a Birthday's present.

(98) Peter sent a Christmas present to his girlfriend, and John did to his mother.

(99) John gave Bill a lot of money, and Mary will Susan.

(100) *John gave Bill a lot of money, and Mary will a lot of attention.

All the pseudogapping sentences above have only one reading, that one in which the coordinator connects two IPs—in which the elided clause is interpreted as a coordinated IP. This is expected since in these cases an auxiliary is present, which forces IP attachment. Once again, even though the coordinator and the subject DPs in each sentence (*Mary* in (99) and (100)), or *John* in (97) and (98)) are attached low, they are reanalyzed and attached high as soon as the auxiliary is encountered. Thus, the auxiliary forces this high attachment and it only allows for one reading of the sentence—that one in which the DP following the coordinator is interpreted as the subject of the second conjunct.

5.2.8 Summary

I have presented an ellipsis resolution routine, which takes into consideration efficiency and economy considerations, and which applies minimalist operations in the way defined by Weinberg (1999). I have accounted for the different locality effects in ellipsis as a result of (i) the auxiliary effect, (ii) low initial attachment of coordinates, and (iii) multiple spell-out operations which render syntactic structure unavailable.

With a two-fold process like the one I have defined for VPE, where certain structure is computed on-line and interpretation of the elided VP carried at LF, we can account for some ellipsis examples that an algorithm like that of Lappin & McCord (1990) cannot. To be precise, Lappin and McCord's algorithm cannot account for cases where the gap precedes the antecedent (as in (101)) or examples of ellipsis in discourse (as in (102)), since their algorithm depends on accessing the antecedent to assign structure to the gap and to interpret it.

(101) Although Mary didn't, Peter went to the party.

(102) A: Who came?

B: I did.

Our algorithm, in both cases, will predict a VP on-line (a top-down prediction from the IP), and will access the antecedent at LF. Even though in both cases there is not a c-command relation between the antecedent and the gap (in (101) the gap precedes the antecedent and in (102) they belong to different sentences), this is not a problem, since at LF access to both material that is in the c-command path and outside the c-command path is possible.

In the next section, I discuss other cases that the algorithm I defined can also account for: cases of strict and sloppy readings and of interaction between Binding Theory and Quantifier Raising.

5.3 More on Interpretation at LF

5.3.1 Computing C-command Relations.

5.3.1.1 C-command

One of the most fundamental concepts in minimalist grammar, basic for defining binding relationships, is the concept of c-command, defined by Epstein (1999) as follows:

(103) C-Command: α commands all and only the terms of the category β with which α was paired by Merge or Move in the course of the derivation.

C-command is a product of the operation merge. It needs to be computed for the binding of anaphors, pronouns, and traces. C-command is also going to play an important role for ellipsis.

The definition above envisions c-command in the context of the bottom up derivations. This definition also applies to parsing. How is c-command computed in parsing? Recall from our discussion of Weinberg (1999) in section 5.2.1 that c-command is deduced from precedence relations (in the spirit of Kayne's (1995) LCA)—whenever this precedence relation among terminals cannot be maintained (whenever a word cannot be merged with the existing structure) spell-out applies (as in Uriagereka (1999)). Consider sentence (104) below, and the command relations among terminals:

(104) [$_{TP}$ [$_{DP}$ Which picture of himself$_i$]$_k$ did John$_i$ like t$_k$]?

Multiple Spell-Out gives us the basic command units in the sentence, that is, the basic blocks of structure where the elements stand in a command relation. So, in sentence (104), MSO tells us that the DP *which picture of himself* is a command unit in itself, since it is spelled-out separately, and attached to the mother command unit (the TP). The command units for sentence (104), are represented in (105) (the wh-constituent DP command unit), and in (106) (the TP mother command unit). In both, the arrow below indicates the direction in which c-command applies—every single element in the list commands every element that follows:

(105) **DP$_k$** {D-which NP {N-picture PP {P-of DP {D-himself}}}}

– ➤

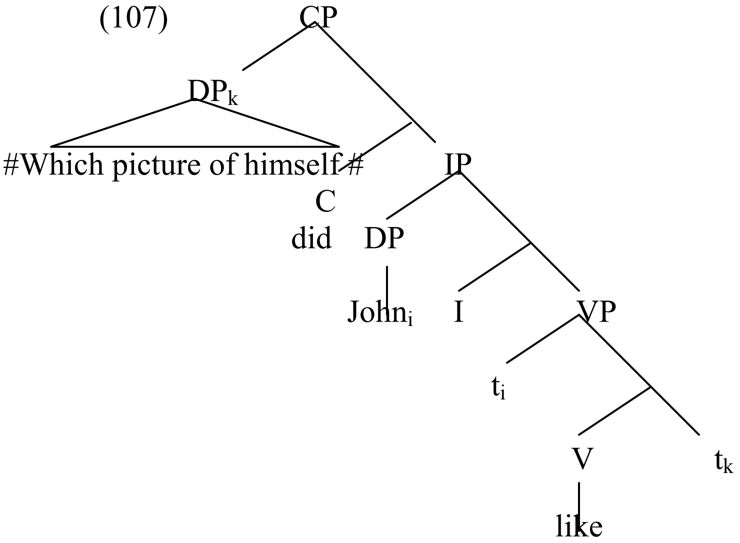
C-Command

(106) CP {**DP$_k$** C-did TP {DP$_i$-John T VP {trace$_i$ V-like trace$_k$}}}

– ➤

C-Command

(107)



There is no command relation established between the elements inside the DP *which picture of himself* and the rest of the elements in the structure (in the mother command unit). But we do know that that DP node c-commands the lower position with which it is associated (t_k)—since they belong to the same command unit (that one in (106) above) and the DP precedes the trace position.

Thus, MSO accounts for those command relations existing between the items in the sentence with respect to what I have called before their "grammatical position", or in other words, their "surface" position (meaning the position where they are overtly realized).

5.3.1.2 Derived C-command

Now, I would like to call the reader's attention to the following: the anaphor *himself* in sentence (104) is bound by *John*; they are coindexed. In order to be bound, an anaphor must be c-commanded by its binder, but how can *John* c-command the anaphor? Observing the tree in (107) above, and the sentence command units above, we can conclude that *John* cannot bind *himself* in the specifier of CP. If command is deduced from precedence, *John* cannot command the anaphor since it does not precede it, and what is even worse, the two elements belong to separate command units, so no command relation exits between them.

I would like to bring back into the discussion the notion of Chain (this is going to help us answer the question of how command is computed for the anaphor binding). I would like to remind the reader how it was proposed that whenever a Move operation takes place a Chain is built. Suppose that whenever one of these chains is built a register is created where information about the links of the chain is stored—i.e., information about the two positions with which a syntactic object is related. Suppose also that every time a node is postulated it receives a variable name—such as: X1, X2, …, XN—which can be used any time we need to refer back to it; so that a node can be addressed by its variable name rather than by a description such as “the DP in the specifier of CP”. Thus, every chain is stored in a register in the following manner:

(108) Chain: [XP P1 P2]

XP stands for the phrase (the syntactic object) that is being moved. P1 and P2 stand for the two links of the chain, or in other words, for the two positions in the tree with which the syntactic object is related. For example, for sentence (104) above the following chain will be registered (suppose that the DP in [spec, CP] receives the variable X1 and that the trace position receives the variable name X10):

(109) Chain: [DP{which picture of himself} X1 X10]

By building this chain we relate a syntactic object to two positions in the tree—even though it is only overtly realized in one place (X1) it is related to (it exists in) two (X1 and X10).

For the purposes of the binding of the anaphor *himself* in sentence (104) we are interested in the relationship that exists between the DP subject *John* and the trace position with which the wh-element is related (t_k). This information can be extracted from the command units in (105) and (106) above. Now, does the DP *John* c-command X10? The DP *John* precedes the trace position and they belong to the same command unit, so it c-commands it.

But, what about the relation between *John* and the anaphor *himself*? To answer that question let us go back to the definition of c-command in (103) above. Let the DP *John* be α and the I node be β. The DP commands I, and all its terms, therefore it commands *himself*.

Sentence (110) below, an example of VPE, also raises a similar question. Recall that two readings are available for this sentence, which can be paraphrased as follows: (i) John shaved himself before Peter shaved him, and (ii) John shaved himself before Peter shaved himself:

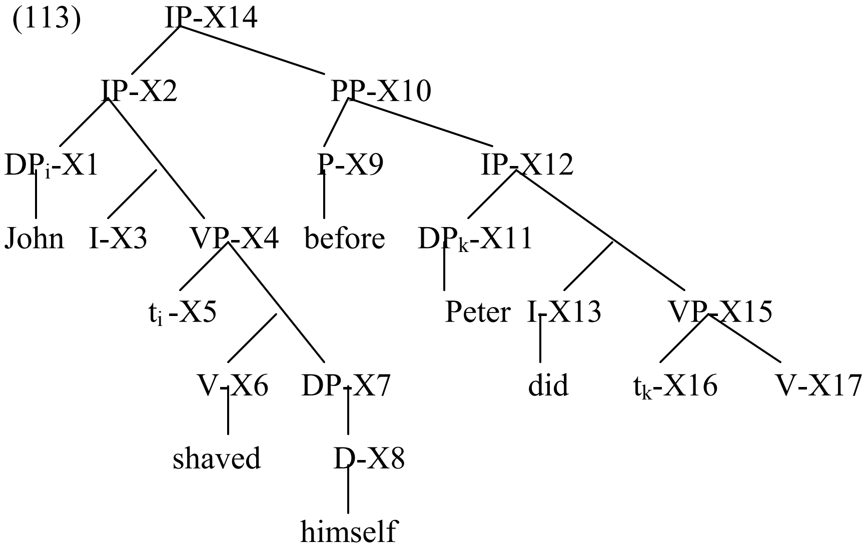
(110) John shaved himself before Peter did.

Consider the reading in (ii); the anaphor *himself* here also must be c-commanded by the antecedent *Peter* so as to be bound by it. MSO will give us the basic command relations, in (111) and (112). Again, we need to store the positions with which the VP *shaved himself* is associated, so that the command relations between *Peter* and the anaphor in its lower position can be computed:

(111) IP-X2 { DP$_i$-X1:John I-X3 VP-X4 { t$_i$-X5 V-X6:shaved DP-X7
{ D-X8:himself }}}

(112) IP-X14 { IP-X2 PP-X10 { P-X9:before IP-X12 { DP$_k$-X11:Peter
I-X13:did VP-X15 { t$_k$-X16 V-X17 }}}}

(113)



(114) Chain: [VP{shaved himself} X4 X15]

In order to get the reading in (ii), the anaphor *himself* has to be bound by the DP subject *Peter*; and in order to be bound it has to be c-commanded. Does the DP c-command the anaphor in VP-X15 (the lower link of the chain)? The DP c-commands the I (the category with which it was merged), therefore it c-commands the VP, and also the DP anaphor.

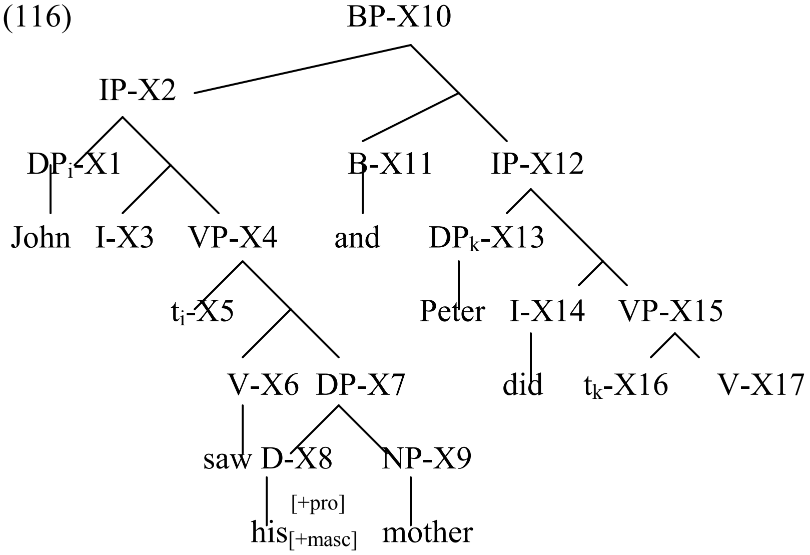
5.3.2 How the Minimalist Algorithm Accounts for Strict and Sloppy Readings

The idea of derived c-command that I have just presented above together with the analysis of Murguía (2000)—which hinges on the distinction between mere phrasal categories versus phrasal categories with lexical material—can be applied to get the sloppy and the strict readings with the algorithm presented so far.

Let us start by discussing an example to see where this fits with the material discussed previously. We also show how both readings are obtained. Consider sentence (115), which will be assigned the structure in (116):

(115) John saw his mother, and Peter did too.

(116)



(117) **IP-X2** { DP$_i$-X1:John I-X3 VP-X4{ t$_i$-X5 V-X6 DP-X7{ D-X8:his NP-X9:mother }}}

(118) BP-X10 { **IP-X2** B-X11:and IP-X12{ DP$_k$-X13:Peter I-X14:did VP-X15 { t$_k$-X16 V-X17}}}

In (117) and (118) above we have the two CUs that are sent to LF. At the LF level the antecedent is accessed and the Chain in (119) is created. This chain, as I mentioned, represents the two positions with which we relate the syntactic structure of the antecedent VP:

(119) [[$_{VP}$ saw his mother] X4 X15]

The lexical structure of the antecedent in (41) is retrieved, and used to interpret the elision, as in (121):

(120) (see (args Agent DP-X1:John
                    Theme DP-X7: his mother))

(121) (see (args Agent DP-X13:Peter
                    Theme @X7))

Binding theory applies, and the pronoun in the first conjunct is interpreted as referring to the subject *John*.[16] When interpreting the theme argument for the elided verb phrase two things can happen (i) when going back to X7 (the DP in the antecedent) all the material in that syntactic structure is borrowed, including the lexical material and the index, or (ii) only the categorial information of X7 is considered and interpreted in the gap position. If option (i) is chosen, the strict reading comes out. If, on the contrary, option (ii) is chosen then the sloppy reading obtains. For the sloppy

[16] If the pronoun is interpreted as referring to a third person (e.g. Tom), then the strict reading will be obtained in the same way, and of course the pronoun in that case will also refer to *Tom*.

reading to obtain, the categories borrowed for the antecedent need to be interpreted in the gap position; that is, in the case of sentence (115) the pronoun has to be bound by the subject *Peter*. In order to do that, the DP *Peter* must c-command the pronoun. At this point we need to go back to the notion of derived c-command that was discussed above, and ask the question: Does Peter c-command the VP-X15? The answer is that it does and consequently it c-commands the DP and the pronoun.

5.3.3 Quantifier Raising and Binding Theory Interactions

In the previous chapter, I introduced some examples of ACDs that were problematic for an s-structure VP anaphora resolution algorithm such as the one presented by Lappin and McCord (1990), since operations like QR (taking place at the LF level) affect structural relationships and consequently binding possibilities.

Let us briefly repeat the argument for the sake of clarity, and then explain how the minimalist algorithm proposed here will account for those cases. Fiengo and May (1994) observe that an s-structure based theory cannot account for sentences like (122) and (123):

(122) *Mary introduced John$_i$ to everyone that he$_i$ did.

(123) Mary introduced John$_i$ to everyone that his$_i$ mother did.

Both sentences respect Principle B of the Binding Theory, however, only (123) is grammatical. We can distinguish these two cases at the LF level. Consider (124) and (125) below, the structure of the sentences after QR has applied:

(124) *[$_{\text{IP}}$ everyone that he$_{i}$ [$_{\text{VP}}$ **introduced John to t]** [$_{\text{IP}}$ Mary introduced John$_{i}$ to t]]

(125) [$_{\text{IP}}$ [everyone that his$_{i}$ mother [$_{\text{VP}}$ **introduced John$_{i}$ to t]** [$_{\text{IP}}$ Mary introduced John to t]]

After QR, the structural relationships between the pronoun and the name are changed. Now we can account for the differences between the two sentences above. Sentence (122) at LF has the structure in (124); the pronoun *he* c-commands the name *John* and this violates Principle C. On the contrary, sentence (123), which has the structure in (125) at LF respects Principle C since the embedded pronoun does not c-command the name.

How can we account for these facts with the proposal I am defending? I am going to start by discussing an example of ACD where no interaction between QR and binding theory occur—to see how QR by itself is accounted for—and then move to an example, where interactions are observed.

5.3.3.1 A Simple Example of ACD: No BT and QR Interaction

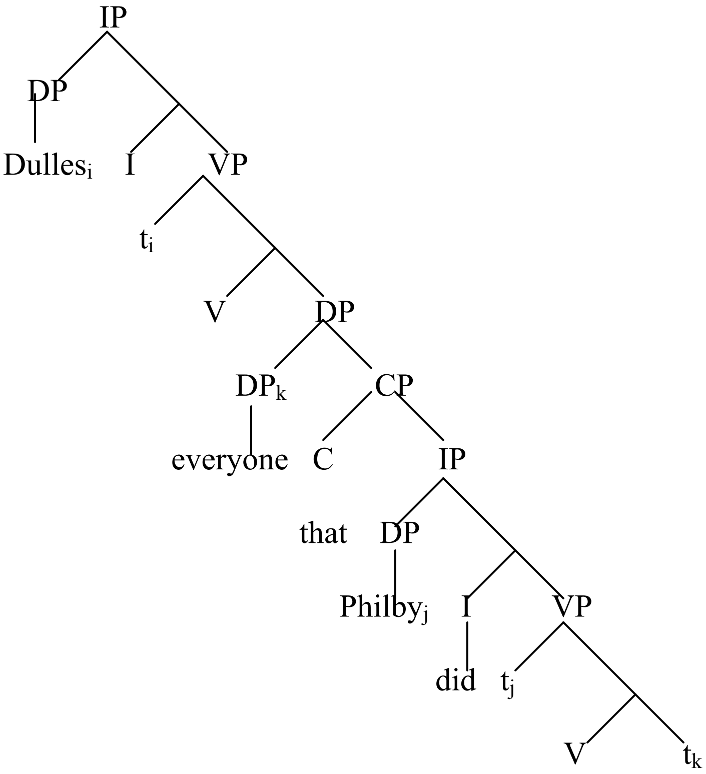
Let us assume that once we have the basic command units that are sent to LF, these can be manipulated there. Thus, QR will be manipulating the c-command units that have been sent to LF.

In sentence (126) below, after all the input words have been parsed, a tree of the sort in (128) below is built and the information about CUs in (127) is passed on to the LF level. QR involves taking the quantified phrase and raising it to take scope over the whole sentence. After QR applies the command relations are affected, the new command units are those in (129) and (130):

(126) Dulles suspected everyone that Philby did.

(127) IP [DP:Dulles$_i$ I VP [t$_i$ V:suspected DP [DP:everyone$_k$ CP [C:that
IP [
DP:Philby$_j$ I:did VP [t$_j$ V t$_k$]]]]]]

(128)



(129) **DP$_n$** { DP:everyone$_k$ C:that IP { DP$_j$:Philby I:did VP { t$_j$ V t$_k$ }}}

(130) IP { **DP$_n$** IP { DP:Dulles$_i$ I VP { t$_i$ V:suspected t$_n$ }}}

Binding theory applies to the structure resulting from QR. BT will take into account that there are two CUs and that the elements which belong to the CU in (129) do not interact with (do not command) those terminals in (130). Now that the idea of how QR can be accounted for in this theory has been introduced, let us discuss one example, where QR affects binding theory results.

5.3.3.2 One Example of QR and BT Interaction

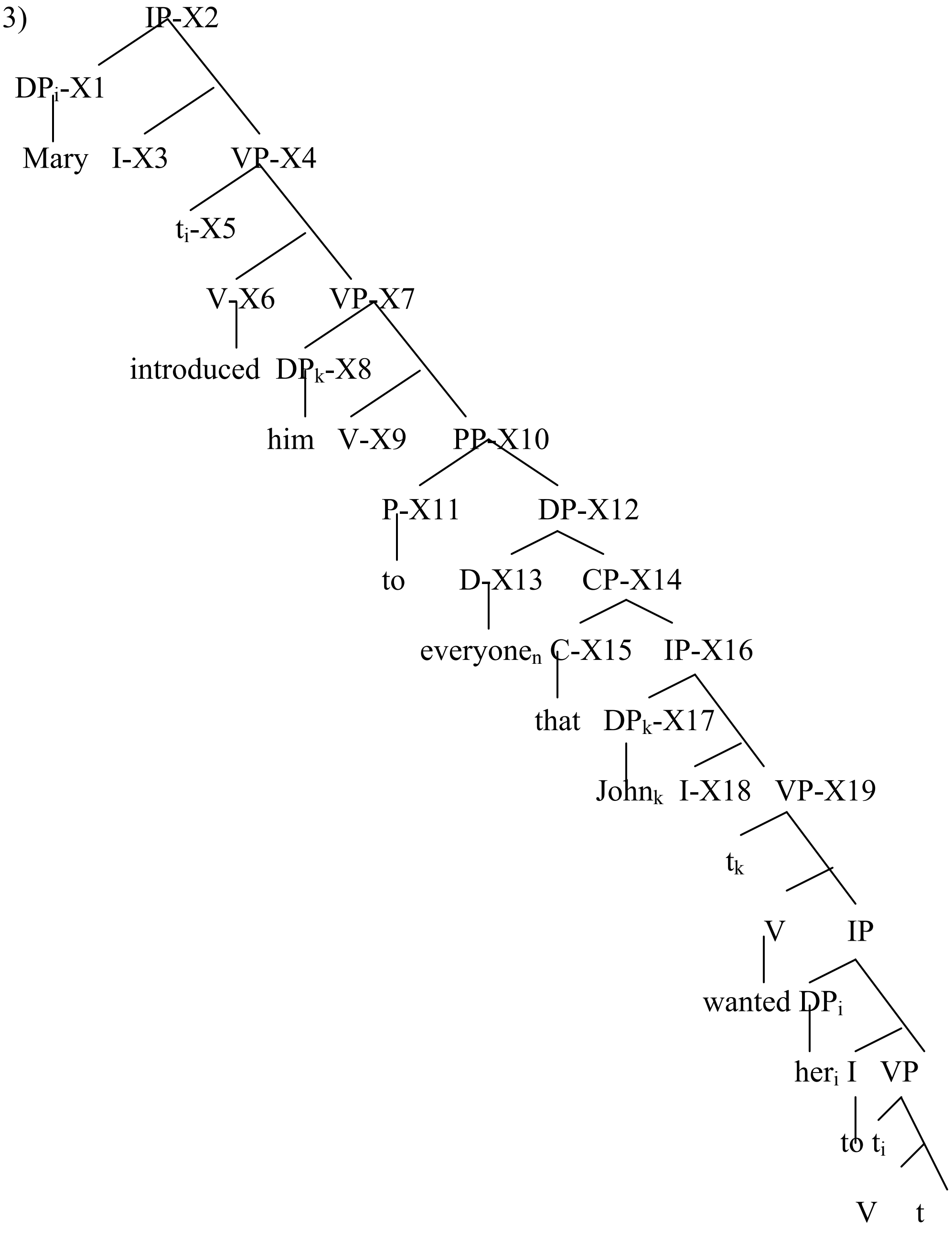
Consider sentence (131) now. According to the overt structure of this sentence, it should be ungrammatical, since it violates Principle C of the binding theory—the pronoun *him* that is coindexed with the name *John* c-commands it. In other words, if we take the precedence order of the terminal elements as the relevant structure to which binding theory applies, the wrong results are obtained. I have already argued, following Fiengo and May (1994) that it is at LF where binding theory should apply.

(131) Mary introduced him_i to everyone that John_i wanted her to.

After the whole sentence is parsed the structure obtained is that one in (133), and the command units are in (132), the pronoun *him* c-commands the DP *John*:

(132) IP { DP:Mary_i I VP { t_i V:introduced VP { DP:him_k V PP { P:to DP {
DP:everyone CP { C:that IP { DP:John_k I VP { t_k V:wanted IP {
DP:her_i I:to VP { t_i V t }}}}}}}}}}

(133)



Once QR applies to this object, the command units are altered, and consequently the command relations among terminals are affected. After QR these are the commands units for the sentence above:

(134) **DP**$_n$ { DP:everyone CP { C:that IP { DP:↙John$_k$ I VP { t$_k$ V:wanted IP { DP:her$_i$ I: to VP { t$_i$ V t}}}}}}

(135) IP { **DP**$_n$ IP { DP:Mary$_i$ I VP { t$_i$ V:introduced DP: ↙him$_k$ PP { P:to t$_n$}}}}

As can be observed in (134) and (135) above, the command relation between the DP *John* and the pronoun *him* is changed after QR—the pronoun does not c-command the name any longer (they are in separate command units), and Principle C is respected.

To conclude, let us say that with the two-fold process that I have proposed for ellipsis resolution—a two-fold process where some work is done on-line and some at the ÑLF level—we cannot only account for several ellipsis phenomena (VPE, pseudogapping and gapping) and the difference with respect to locality effects among them, but also account for the strict and sloppy readings, as well as for the interactions at the LF level of BT and QR.

5.4 Conclusion

I have offered an account for parsing elliptical constructions which makes use of the minimalist operations: Merge, Move, and Spell-out; which takes into considerations efficiency and economy issues, and which makes use of local information.

I have accounted for the presence/absence of locality restrictions as a result of overt tense presence/absence, and of the availability of left context, which in turn is a consequence of (i) low initial attachment of coordinates, and (ii) spell-out operations which render syntactic structure unavailable. We have seen that in the case of gapping the antecedent needs to be accessed to assign structure to the gap, therefore locality restrictions apply. While on the case of VPE and Pseudogapping, a VP may be predicted without resorting to the antecedent which is only accessed for interpretation purposes. Therefore, they are not subject to locality.

In the case of VPE and Pseudogapping the gap is interpreted in two steps: (i) building on-line only the minimal amount of structure to accommodate input items, bind traces, and satisfy grammatical constraints, and (ii) accessing the antecedent at LF for interpretation purposes.

We have also seen that some of the advantages of such a two-fold process are not only accounting for locality restrictions, but also for both antecedent-gap and gap-antecedent cases, as well as for ellipsis in discourse and interactions between Binding Theory and Quantifier Raising in ACD cases. Thus, we need to conclude that, contrary to what Lappin and McCord (1990) suggest, the LF level plays a role in ellipsis resolution: there are certain operations that take place at LF, and all the work cannot be done on-line.

CHAPTER 6: AN ALGORITHM FOR A MINIMALIST PARSER

In this Chapter, I introduce a parser that is defined on a minimalist grammar. I also provide a general algorithm definition for the parser. In order to do so, I take Weinberg's (1999) model of the human sentence processing introduced in the previous chapter (Merge, otherwise Move, otherwise Spell-out) with the extensions proposed there to deal with coordination and ellipsis, and I embed it in a computational model of the sort described by Pullman (1986).

I start by introducing Pullman (1986), who describes a context-free parsing algorithm which basically works bottom-up, but which is also capable of making left-corner predictions. After this, I introduce the minimalist parser I propose and certain modifications to Pullman's model that will allow us to build syntactic structure incrementally, and to account for movement operations, as well as for coordinates and ellipsis.

6.1 Pullman (1986): Grammars, Parsers and Memory Limitations

Pullman (1986) defines a parser based on a Shift-Reduce algorithm (Aho and Ullman 1972), but which is able to deal with incomplete constituents, along the lines of the left-corner parsers discussed by Johnson-Laird (1983), Abney & Johnson (1991), and Resnik (1992). It operates non-deterministically, backtracking when necessary, and finding all possible parses for a sentence.[1] It works from left-to-right processing each word as much as possible.

Sentence input words are shifted to and assembled together in the stack, which can be defined as a list of entries. Each entry represents a wholly or partially recognized constituent and is a triple of the sort in (1) below. The first element of the triple is the category label of the constituent that the entry stands for. The second represents the “needed” list of constituents that must be found to complete the current category. And the third is the semantic interpretation of the category:

(1) <category, needed, interpretation>

6.1.1 Parser Operations: Shift, Invoke-Rule, Combine, and Clear

There are four basic operations: Shift, Invoke-rule, Combine and Clear. The decision of which one to apply depends on the state of the parse. If there is more than one possibility, then all apply creating one configuration for each on the parser’s agenda. Shift takes the next input word and creates a new agenda entry for each sense that this word has in the dictionary. For a word like *man*, this operation will create an entry on the stack which contains a triple like that in (2) below:

[1] Pullman himself acknowledges that this is not realistic; he proposes that the main loop of the parser should include a call to a routine that chooses among different possibilities whenever two or more parses are predicted.

(2) <N, nil, MAN>

Invoke-rule applies to complete constituents on top of the stack by checking the rules of the grammar to see whether that constituent could begin another higher level constituent. If the category of the entry on top of the stack matches the first element of the right-hand side of a rule, i.e. the rule's left corner, then the constituent that rule represents is predicted by creating a new entry in the stack. So, if we have a constituent of the sort in (3) on top of the stack, and a rule like the one in (4), then a new constituent can be predicted (an NP), and a new entry (see (4)) added to the stack in substitution of (3):

(3) <**Det**, nil, EVERY>

(4) NP→ **Det** N[2]

(5) <NP, N, λn [EVERY (n)]>

The category (NP) of the new entry in (4) matches the constituent on the left-hand side of the rule that has been invoked, and the needed list (N) is formed by those elements on the right-hand side of the rule, except for the *Det*. Pullman links this operation of Invoke-rule to lookahead of one word, in order to avoid false starts in ambiguity cases, such as coordination and optional modifiers. However, we will see

[2] Pullman assumes a non-DP analysis here, where the determiner *The* is part of the NP phrase rather than the head of a DP.

when we discuss some cases later on that this lookahead is not enough for ellipsis, since it is not until the auxiliary is recognized that the ambiguity of the correct attachment site for the coordinator disappears.

The operation Combine combines two constituents: a complete one (on top of the stack) and an incomplete one below it; only if the category of the former matches the needed category of the incomplete one. If the stack contains the following elements in (6), then Combine can apply creating a new entry, as in (7):

(6) <**N**, nil, MAN>

<NP, **N**, λn [EVERY (n)]>

(7) <NP, nil, EVERY (MAN)>

Finally, the operation Clear applies when a complete or completable representation of a proposition has been built. The purpose of this operation is to prevent the stack from growing unboundedly when parsing right-branching structures of the sort in (8); thus, preventing short-term memory problems. The syntactic structure disappears, but the semantic representation is still available for the rest of the parse. Pullman sets the threshold for Clear to 6 elements in the stack, though he admits that its exact nature is quite complex and that it might be influenced by linguistic and non-linguistic factors. If Clear applies to a configuration like that in (9), the result is (10):

(8) Mary said that she believes the claim that Peter promised he would go.

(9) … top of the stack …

<**VP**, S, λs [HOPES (s)].

<S, **VP**, λvp [vp (SOME (MAN))]>

… bottom …

(10) … top of the stack …

<S, S, λx [λvp {vp (SOME (MAN))} [λs {HOPES (s)} (x)]]>

6.1.2 One Parse Example

Let us now consider one example with the operations described above that Pullman himself discusses. Looking at a simple case allows us to better understand the proposed changes made by our model. For a sentence like that in (11), using a sample grammar like that in (12), the parser operations are those in (13). Each step is indicated with a Roman numeral, together with the operation that applies annotated to the right:

(11) The cat caught the mouse.[3]

(12) S→ NP VP

NP→ Det N

VP → V NP

[3] For this and all the examples that Pullman discusses, his illustrations assume that the parser is making the right choice when more than one possibility is available; the parsing algorithm that he offers is of a non-deterministic nature.

(13)	(i)	<Det, nil>	Shift: The
	(ii)	<NP, N>	Invoke-rule: NP→ Det N
	(iii)	<N, nil>	Shift: cat
		<NP, N>	
	(iv)	<NP, nil>	Combine
	(v)	<S, VP>	Invoke-rule: S→ NP VP
	(vi)	<V, nil>	Shift: caught
		<S, VP>	
	(vii)	<VP, NP>	Invoke-rule: VP → V NP
		<S, VP>	
	(viii)	<Det, N>	Shift: the
		<VP, NP>	
		<S, VP>	
	(ix)	<NP, N>	Invoke-rule: NP→ Det N
		<VP, NP>	
		<S, VP>	
	(x)	<N, nil>	Shift: mouse
		<NP, N>	
		<VP, NP>	
		<S, VP>	

(xi)	<NP, nil>	Combine
	<VP, NP>	
	<S, VP>	
(xii)	<VP, nil>	Combine
	<S, VP>	
(xiii)	<S, nil>	Combine

I would like to call the reader's attention to steps (v-xiii) above. After the subject NP *The cat* is recognized and a sentence predicted by invoking the rule S→ NP VP (in step (v)), the algorithm proposed by Pullman does not build syntactic structure incrementally: it builds the rest of the constituents in a strictly bottom-up manner. It goes on recognizing the verb (in (vi)) and predicting a VP by invoking rule VP → V NP. However, the NP that the category VP needs is not combined with the VP until that NP is completed. Only when the NP *the mouse* is completed (in (xi)), is it combined with the VP (in (xii)), which in turn, after completion is combined with the category S (in (xiii)).

In the example above, since the structure of the object NP is not too complex, this constituent is combined with the rest of the structure a few steps after the category NP is predicted; and the VP is combined with the NP subject, immediately after. However, in a sentence with a more complex object it will take many more steps to attach it. It has been argued in the psycholinguistics literature that humans process sentences in an incremental manner, and in particular, that a category's thematic function in higher structure is predicted before the category is complete. This suggests that some attachment decisions must be made before the category is complete.

6.2 A Minimalist Version of Pullman's Parser

The parser that I am about to describe is a non-deterministic serial parser—like that of Pullman—which allows backtracking and reanalysis of structure assigned to a sentence. The general algorithm is described in section 6.2.4 below. Any time that there is a choice between different structures (i.e. different phrasal rules), the point at which the search path diversifies is marked, so that the parser can backtrack to that point and reanalyze if necessary.

For illustrative purposes, consider a sentence like that in (14). After the determiner is shifted to the stack (see (15) step (i)), the next step is to Invoke-rule. Assuming we have a grammar like that in (16), there are two possible rules that can be invoked, represented in step (ii). At this point, the parser makes a choice, and marks this point as the backtracking point. If the parser chooses the rule DP → D, it will ultimately fail and backtrack to this point to try to assign the other available structure to the sentence.

(14) That teacher from England gave a talk.

(15) (i) <D, nil> Shift: that

(ii) <DP, nil> **DP → D**

(iii) <IP, {I, VP}> Invoke

(iv) <N, nil> Shift: teacher

<IP, {I, VP}>

FAIL!

(ii) <DP, NP> **DP→ D NP**

(iii) <N, nil> Shift: teacher

<DP, NP>

…

I take Pullman's operations as the base, but I introduce some changes in those so as to deal with (i) the incrementality issue mentioned above, (ii) movement operations, or in other words, displaced constituents, (iii) coordination, and (iv) verb phrase ellipsis.

6.2.1 A Brief Comment on the Framework

Before we consider those modifications in detail or discuss some examples, I briefly comment on some differences between Pullman's grammatical assumptions and the grammatical framework I assume (Minimalist Program (Chomsky 1993, 95, 98)). First, the grammar that guides our parsing algorithm differs from the grammar that Pullman proposes in that it includes both lexical categories (e.g. V, N), and functional categories (e.g. C, T). I assume a more structured representation for sentences. In (16) below, I include the sample grammar that we will be using:

(16) CP → DP I IP

CP → C IP

DP → D

DP → D NP

IP → DP I VP

VP → DP V

VP → DP V DP

VP → DP V PP

NP → N

NP → N PP

NP → N CP

Our grammar also builds a richer syntactic structure: it keeps track of the different positions in a sentence that syntactic constituents can be related with (i.e. the surface position and the base position). To be precise, it keeps track of moved/displaced constituents. Thus, we present a context-free grammar, but the parser does additional bookkeeping to keep track of deep structure (i.e. moved constituents). Consider (17) and (18) below:

(17) Mary read that novel.

(18) [Which novel]$_i$ did Mary read t_i?

In (17), the object *that novel* of the verb *read* is related only to one position in the sentence: its base position. In this case, the surface position of that constituent (i.e. that one in which it is pronounced) and its base generated position (i.e. that one in which it receives a theta-role) are the same. That is not the case in sentence (18), where the object *which novel* can be related to two positions: (i) its surface position, at the beginning of the sentence, and (ii) its base generated position, marked with the trace *t.* To deal with this phenomenon, a new data structure was introduced: Chains (see discussion in Chapter 5, but let us review the main idea here). Everytime a Wh-element or an NP is recognized, a chain is created and stored (see (19), where the structure of Chain is represented). Chains keep information about the moved constituent category, and about the two positions which the constituent is related to: (i) P1 which is the surface/grammatical position (where case or wh-features are checked), and (ii) P2 which is its base-generated position (where theta-roles are assigned):

(19) [XP P1 P2]

Every DP/NP or Wh-element must have case and a theta-role assigned to it, and Chains help us keep track of this information. For a sentence like (20), when the DP *which book* is built, a chain is stored with the DP assigned to XP, and with P1 having been assigned the surface position of the DP. At this point, P2 is still empty.

(20) [Which book]$_i$ did you recommend t$_i$?

After the verb is parsed, P2 will be assigned the position of the trace: there is no overt element to fill the position of the object of the verb, but we have a DP *which book* which needs to get a theta-role; we can access this type of information by checking the chains that have been built. By virtue of being related to the lower position (that of the object) it receives a theta-role.[4] This addition completes the components of the general architecture.

6.2.2 General Architecture

In the chart below, I represent the general architecture of the parser I am going to describe. There is (i) a stack to which input items are shifted, and where they are combined, (ii) a chain store, where information about the chains that are built during parsing is kept, and (iii) a constituent store, where information about the syntactic structure is maintained and updated. See Figure 1, below:

[4] I do not intend to develop a detailed account of how features are assigned/checked by the parser in the present work. For the present purposes, it is just important to keep in mind that DPs/NPs need to have Case and a Theta-role, and that chains help us in storing and retrieving which category is complete or not in that sense.

I just want to give a taste of what the general idea is, and suggest a way in which it could be implemented. The general idea is that (i) categories enter the stack with feature variables that need to be assigned a value (e.g. Case, which will need to be assigned the value Nominative or Accusative), and (ii) there are certain positions in the syntactic structure where features are assigned (e.g. in the specifier of CP, the wh-feature of a wh-element can be checked, or in the specifier of T (or I) Nominative case is assigned).

One way in which this system could be implemented is by creating entries with information not only about the category of the input item and the list of needed categories, but also about the features that need to be checked (e.g. the entry for a word like who would be: <D, {Case, Theta-role, Wh-feature}, nil>; or for a word like man: <N, {Case, Theta-role}, nil>). Then, these features could be assigned if we augment our grammar rules with constraints that need to be satisfied in order for a rule to apply (e.g. a rule like VP→ DP V will be augmented with the condition of assigning Nominative case to the DP).

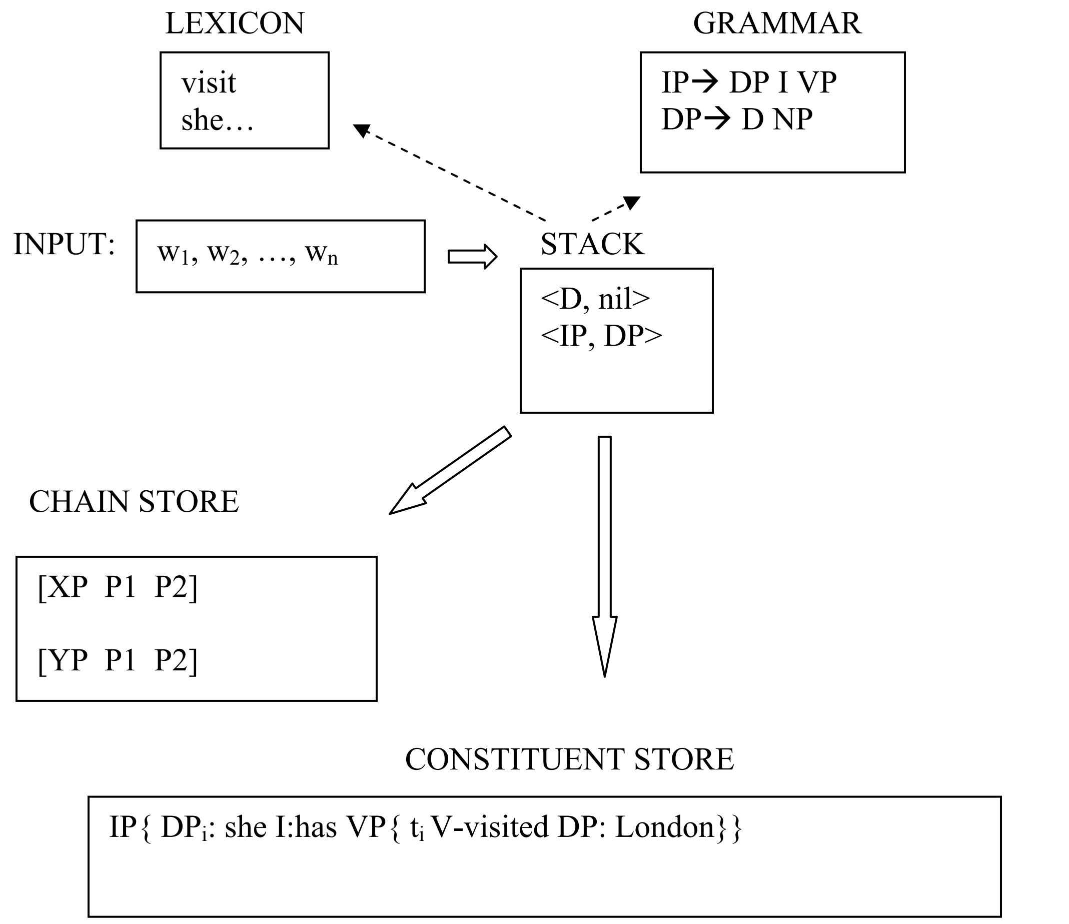


Figure 1

The parser has access to the Lexicon, which is checked every time an input word has to be shifted to the stack, in order to check its category specifications, and other information (e.g. subcategorization information for verbs). The Grammar is also accessed by the parser so as to combine entries on the stack or to make predictions about new constituents. Information about chains or constituent structure is sent from the stack to the Chain Store and the Constituent Store, respectively. The Chain Store can contain more than one incomplete chain at a time; in that case, chains are retrieved in a Last-in First-out order.

I also add the new data structure "Constituent Store", so that the parser can keep track of the syntactic structure computed for a sentence. Information about the constituents that have been built is kept there; this can be accessed, and constituents can be retrieved in order to reanalyze the syntactic structure assigned to the sentence. If spell-out (one operation that our parser includes and that I will explain in the next section) affects one of these constituents, then its syntactic structure is not available any longer. I represent spell-out material between the symbols # #, following Uriagereka (1999) (e.g. #DP#).

6.2.3 Some Modifications to the Parser Operations

6.2.3.1 Combine Revisited: Building Syntactic Structure Incrementally

I propose that, in order to deal with the incrementality issue, Pullman's Combine operation needs to be modified; so that it allows combining not only a complete constituent on top of the stack with an incomplete one below it, but also an incomplete constituent with the incomplete one below it.[5] With Combine defined in these terms, syntactic structure can be built in an incremental way. This modified

[5] Clear allows combining elements in both circumstances, but it only applies to categories V or S, and we want this operation to apply across the board.

operation applies both to the configuration in (21) (what Pullman already claims), as well as (21), giving as a result (22) and (22), respectively:

(21) a. < **PP**, nil> b. <**PP**, NP>
<NP, **PP**> <NP, **PP**>

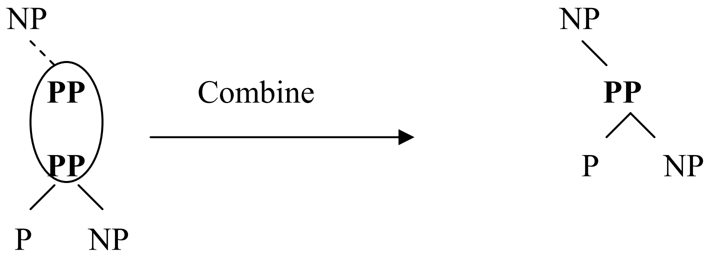
(22) a. <NP, nil> b. <NP, NP>

In (21), since the PP constituent on top of the stack is not complete (it needs an NP), Combine transfers that needed category to the list of needed elements of the NP constituent with which it combines the PP, as shown in (22) (the transferred NP appears underlined). I include the general description of this modified operation in (23) below:

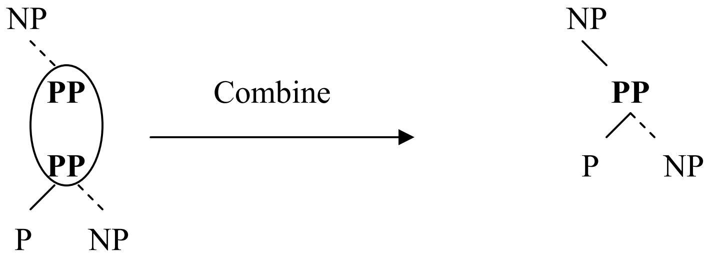
(23) IF we have a configuration of the following sort in the stack
a. <**XP**, nil> b. <**XP**, ZP>
<YP, **XP**> <YP, **XP**>
then after COMBINE applies the stack looks like
a'. <YP, nil> b'. <YP, ZP>

The trees in (24) and (25) show the difference between our operation of Combine and Pullman's. The dotted arcs represent top-down predictions of nodes, which need to be verified bottom-up. In (24), a PP is predicted top-down (represented by the dotted arc), a PP is also built bottom-up. Not until the PP is complete (i.e. all its daughter nodes are recognized) is it combined with the NP. In (25), the same reasoning applies, with the difference that when the PP and the NP are combined the bottom-up built PP is not complete—it still needs an NP, which has been predicted (represented by the dotted arc), but not verified bottom-up:[6]

(24)



(25)



Consider how sentence (11) above (repeated here as (26)) is parsed with this modified Combine operation. The steps are those represented in (27) (I use Pullman's grammar for this example, rather than the minimalist grammar I introduced in (16), so that the comparison between the operations the parser performs before and after modifying

[6] For a complete discussion of these matters (top-down predictions and bottom-up confirmations, and operations that allow to combine incomplete constituents) I refer the reader to (i) Johnson-Laird (1983), where left-corner parsing is discussed, (ii) Abney & Johnson (1991), who discuss left-corner arc-eager parsing strategies (i.e. as soon as a node is introduced it is attached), and (iii) Resnik (1992), who discusses both proposals and suggests a translation of arc-eager parsing into a composition function (in the style of Combinatory Categorial Grammars (CCGs)).

the Combine operation can be clearly established; nevertheless, the argument goes through in the same manner if we used that other grammar). Compare (27) with (13) above. The steps (i) to (vii) would be the same: the subject NP is built and an S predicted, the V shifted, and a VP predicted. However, right after the VP is predicted (step (vii)), the steps the parser takes differ. The new operation Combine allows to merge two incomplete constituents, so the VP can be merged with the category S (even though the VP is not complete yet), and the needed category of the VP is inherited by the S (in step (viii)). The determiner is shifted (step (ix)), and an NP predicted (step (x)). This NP needs an N to be complete, but Combine does not care about that and merges this NP to the S category that is looking for an NP. Finally the N is shifted and combined with the S:

(26) The cat caught the mouse.

(27) (i) …

(v) <S, VP> Invoke-rule: S→ NP VP

(vi) <V, nil> Shift: caught

<S, VP>

(vii) <**VP**, NP> Invoke-rule: VP → V NP

<S, **VP**>

(viii) <S, NP> Combine

(ix) <Det, N> Shift: the

<S, NP>

(x)	<**NP**, N>	Invoke-rule: NP → Det N
	<S, **NP**>	
(xi)	<S, N>	Combine
(xii)	<**N**, nil>	Shift: mouse
	<S, **N**>	
(xiii)	<S, nil>	Combine

Thus, we can see that modifying Combine (in the way described above) allows the parser to attach input words as they are shifted to the stack. Constituents do not have to be complete to be combined with other constituents, thus allowing the parser to build the syntactic structure for a sentence incrementally.[7]

One could claim that in order to account for examples like that one in (26) above, we could just have the operation Clear applying with no threshold restriction, that is, anytime two constituents of type S or V are together in the stack they are combined. This is what Resnik (1986) argues, though the motivation for him is

[7] The same result is obtained in CCGs by a rule of composition. I include below the definition of this operation that appears in Steedman (1990):

(i) Forward Composition:
X/Y Y/Z → X/Z

This rule can be interpreted in the following way: if we have a constituent X that is looking for one of type Y, and a constituent Y that is looking for one of type Z, then the result of composing these two is a constituent X that is looking for a Z.

different.[8] In any case, there are still examples that Clear would not account for, even if it applies across the board (e.g. step (xi) above (since the top entry in (step (x) is not of category S', S, or VP). Consider also sentence (28) or (29) below:

(28) Mary met [$_{\text{NP}}$ that teacher [$_{\text{PP}}$ from England]].

(29) [$_{\text{NP}}$The author [$_{\text{PP}}$ of [$_{\text{NP}}$ that book [$_{\text{NP}}$ that [$_{\text{VP}}$ discusses [$_{\text{NP}}$ philosophy]]]]]] is a friend of John.

Whether Clear applies while building the VP in (28) or not does not make a big difference for incrementality. In (28), if Clear applies after recognizing the verb, and predicting a VP, then the VP will be combined with the S node (or IP node in minimalist terms), but the NP *that teacher from England* will not be built incrementally, since Clear does not have anything to say about combining NP (or DP) and PP constituents. Consider the parser operations after the NP subject *Mary* has been parsed, an S predicted, the verb Shifted, and a VP that needs an NP built (see (30)(i) and following steps):

(30) (i) <VP, NP>
<S, VP>
(ii) <S, NP> Clear

[8] Resnik (1986) explains the restriction on center-embedding and subjacency effects by (i) having Clear applying across the board and for categories S', S and VP, and (ii) by developing a notion of interference of different degrees among entries in the stack.

(iii)	<Det, nil>	Shift: that
	<S, NP>	
(iv)	<NP, {N, PP}>	Invoke-rule
	<S, NP>	
(v)	<N, nil>	Shift: teacher
	<NP, {N, PP}>	
	<S, NP>	
(vi)	<NP, PP>	Combine
	<S, NP>	
(vii)	<P, nil>	Shift: from
	<NP, PP>	
	<S, NP>	
(viii)	<PP, NP>	Invoke-rule
	<NP, PP>	
	<S, NP>	
(ix)	<N, nil>	Shift: England
	<PP, NP>	
	<NP, PP>	
	<S, NP>	
(x)	<NP, nil>	Invoke-rule
	<PP, NP>	
	<NP, PP>	

	<S, NP>	
(xi)	<PP, nil>	Combine
	<NP, PP>	
	<S, NP>	
(xii)	<NP, nil>	Combine
	<S, NP>	
(xiii)	<S, nil>	Combine

Not until the last Combine operation applies (step (xiii)) is the NP *the teacher from England* assigned as the object of the VP. Now, consider the steps the parser takes with the modification suggested to the Combine operation:

(31)	(i)	<VP, NP>	
		<S, VP>	
	(ii)	<S, NP>	Combine
	(iii)	<Det, nil>	Shift
		<S, NP>	
	(iv)	<NP, {N, PP}>	Invoke-rule
		<S, NP>	
	(v)	<S, {N, PP}>	Combine
	(vi)	<N, nil>	Shift: teacher
		<S, {N, PP}>	

(vii)	<S, PP>	Combine
(viii)	<P, nil>	Shift: from
	<S, PP>	
(ix)	<PP, NP>	Invoke-rule
	<S, PP>	
(x)	<S, NP>	Combine
(xi)	<N, nil>	Shift: England
	<S, NP>	
(xii)	<NP, nil>	Invoke-rule
	<S, NP>	
(xiii)	<S, nil>	Combine

In this case, the object NP is attached to the structure immediately after it is postulated (step (v)), even though it is not complete. The rest of the input items that complete it are also attached as they are recognized.

6.2.3.2 Invoke Revisited

Another modification to the basic operations described by Pullman that I include is to the Invoke operation. Pullman allows Invoke-rule to predict the constituent on the left-hand side of a rule if the category on top of the stack matches the first element of the right-hand side of that rule (a left-corner prediction). We keep this operation, but I modify it slightly by a call to the spell-out operation; so that precedence /c-command relation is respected (see (32)):

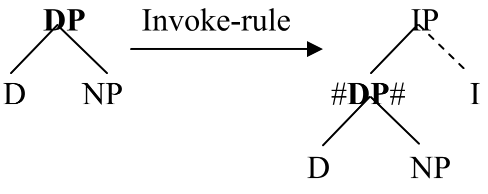
(32) If we have a complete constituent **ZP** on top of the stack <ZP, nil>,
and a rule of the sort XP→ **ZP** X WP
then,
(i) spell-out node **ZP**[9]
(ii) update stack in the following manner: <XP, {X, WP}>

So, for a sentence like that in (33), once a DP has been recognized bottom-up (see (34) step (i)), the rule IP→ DP I VP can be invoked and an IP built (see (34) step (ii)). This is graphically represented in (35) below:

(33) The letter arrived yesterday.

(34) (i) <**DP**, nil>
(ii) <IP, {I, VP}> Invoke-rule: IP→ **DP** I VP

(35)



[9] I include the definition for spell-out in the next section, in example (41).

In addition, I add the option of invoking a rule when the category of the entry on top of the stack matches the head of a rule (see (36)):

(36) If we have a configuration of the following sort on the stack

<**X**, nil>

<YP, XP>

and (i) a rule like: XP → ZP **X** WP

(ii) an incomplete chain with a constituent of type ZP

then,

ZP is assigned a pointer to the constituent that forms the chain, the chain is completed, and the stack is updated in the following way:

<X, nil>

<YP, {X, WP}>

Consider how this operation applies with a sentence like that in (37). We are not going to consider all the steps in detail at this point (to see all the operations see example (55) below). Here we concentrate on the point at which the head-prediction applies, i.e. after the DP subject has been recognized, and an IP that needs a VP predicted (see (39) step (i)). Next, the V is shifted to the stack (step (ii)) and we have at this point an IP that needs a VP (<IP, VP>), and a verb (<V, nil>) on the stack. These two cannot be combined as they stand. The phrase structure rules are checked, to see whether any of them can be invoked. A head-prediction can be made with the rule VP → DP V DP (in (38)). But, in order to combine the V with the rest of the structure, a DP must be recognized first. To fulfill this requirement, the DP subject that was recognized before and was assigned case, but which still needs a theta-role, is related to this position and the chain it belongs to is completed (this information is obtained by checking the chain store). After this Move operation is completed, the stack is updated to that configuration in (iii). Now, the V category can be combined with the IP:

(37) That mother adores her child.

(38) VP → DP **V** DP

(39) (i) <IP, VP>

(ii) <**V**, nil> Invoke-rule: VP → DP V DP

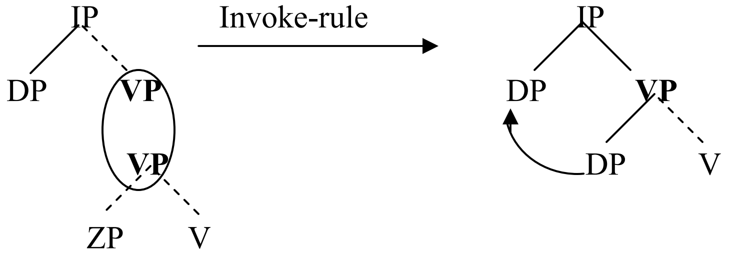
<IP, **VP**>

(iii) <V, nil>

<IP, {V, DP}>

I include also a graphical representation of this operation in (40), which unifies both VPs (the one predicted top-down by the IP and the one predicted bottom-up by the V), and which updates the DP chain by relating the DP subject to the lower position in the tree where it gets its theta-role.

(40)



6.2.3.3 Spell-out Operation

The last operation that our parser includes is Spell-out. This operation applies whenever an input item cannot be merged to the structure that has been built so far. If the input item to be merged cannot be attached to the lowest node of the tree (respecting precedence/c-command), then Spell-out applies.[10] By spelling out constituents that have been built so far, attachment to the tree—to any other node than the lowest one—is made possible, while at the same time maintaining the precedence-command relation among terminals.

I include the general definition of the Spell-out operations in (41) and (42). The first defines spell-out ("lowercase spell-out") which is a subroutine called by Invoke (left-corner prediction) or by Spell-out ("uppercase Spell-out" in (42)). Example (42) includes the definition of the Spell-out routine when reanalysis of syntactic structure is at work. The precise definition of the search function called by Spell-out in (42) is also included in (43):

[10] I would like to the reader to recall that c-command is obtained from the precedence relation among terminals: If A precedes B, then A c-commands B. See discussion in Chapter 5 (sections 5.2.1 and 5.3.1.1)

```
(41)  If spell-out node ZP
      then update constituent store by
            (i)   reanalyzing ZP node as XP dominating ZP, where XP is the
                  left-hand side of the rule XP→ZP X WP
            (ii)  make ZP internal structure unavailable (# #)

(42)  If we have a configuration of the following sort on the stack
        <X, nil>
        <YP, nil>
        and (i) a rule like: XP → ZP X WP
            (ii) no incomplete chain with a constituent of type ZP
      then,
        Search for a constituent of type ZP in the constituent store
          if a match is found:
             then (i)   spell-out node ZP
                  (ii)  update the stack in the following manner:
                        <X, nil>
                        <YP, {X, WP}>
          else  fail![11]
```

[11] This failure refers to the current non-deterministic path, not necessarily to failure of the parse.

```
(43)  Search: Start at lowest node XP in Constituent Store
        Loop: Retrieve the first constituent XP there
                if XP = YP, where YP is constituent being searched for
                        retrieve node
                else go back to loop
        end when reaching the top node
```

Consider sentence (44) below. We are going to see how Spell-out applies in this case. The verb *believe* subcategorizes for either a DP or an IP. The initial analysis for this sentence (see discussion in Chapter 5, section 5.2.1) will take *his mother* as the object of the verb *believe*. (45) shows the structure for the analysis of *John believed his mother*. The structure built so far, and kept in the constituent store is that one in (45), and the stack configuration is represented in (46) step (i). Next step is to shift the auxiliary *is*:

(44) John believes his mother is crazy.

(45) IP {#DP_i:John# I:e VP {DP_i V:believes DP {D:his NP:mother}}}

(46) (i) <IP, nil>

(ii) <I, nil> Shift: is
<IP, nil>

After the auxiliary *is* has been shifted to the stack, the only manner in which this can be attached to the structure is by reanalyzing (neither combine nor invoke in any of its two versions help in this case), so Spell-out is tried (see (42) above). The configuration in the stack is <I, nil> and <IP, NIL> and the rule IP → DP I VP is identified; the constituent store in (47) below is searched for a DP, and a DP *his mother* is found. So (i) the DP is spelled-out, i.e. the constituent store is modified as in (48), the internal structure of the DP is made unavailable (#DP#), and (ii) the stack as in (49):

(47) IP {#DP_i:John# I:e VP {DP_i V:believes **DP** {D:his NP:mother}}}

(48) IP {#DP_i:John# I:e VP {DP_i V:believes **IP** { **#DP#**}}}

DP {D:his NP:mother}

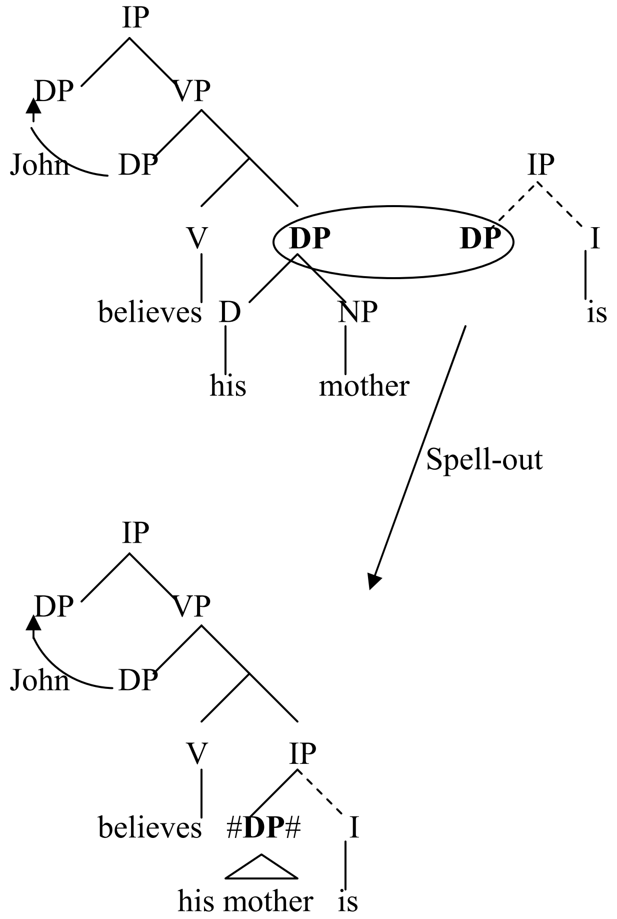
(49) <I, nil> Spell-out DP: IP→ DP I VP)

<IP, {I, VP}>

I represent this graphically in (50), where the DP *his mother*, which is the object of *believe*, needs to be identified with the DP that is predicted bottom-up by the auxiliary in *I*.

(50)

(51)



6.2.4 Algorithm

In this section, I relate the algorithm of human sentence processing Weinberg (1999), repeated in (52) below, to the parser operations we have proposed, which are summarized in (53). I also include a definition of the minimalist algorithm that I define in (54):

(52) A derivation proceeds left to right. At each point in the derivation, merge using the fewest operations needed to check a feature on the

category about to be attached. If merger is not possible, try to insert a trace bound to some element within the current command path. If neither merger nor movement is licensed, spell-out the command path. Repeat until all terminals are incorporated into the derivation.

(53) (i) Shift: Take next input word and create an entry on the stack

(ii) Combine: a complete (or incomplete) entry A on top of the stack with entry B below it, if this needs a constituent of the category of entry A.

(iii) Invoke-rule: (i) left-corner prediction—if the category of the entry on top of the stack matches the leftmost element of the right-hand side of a rule, spell-out that leftmost element and predict the constituent of the left-hand side of the rule, and (ii) head-prediction—if the category of the entry on top of the stack matches the head of a rule and you find an incomplete chain of the appropriate type, then predict the constituent on the left-hand side of the rule.

(iv) Spell-out: (i) spell-out a node XP—update constituent store by reanalyzing XP as YP dominating XP, where YP is the left-hand side constituent of the rule YP→ XP Y ZP you have invoked, and (ii) reanalysis—search for a node of type XP, when you find it spell-out XP.

Now, the question that should be raised is the following: how are (52) and (53) relatable? Here is what I propose. The algorithm in (52) can be summarized as a preference of Merge over Move, over Spell-out. Merge, though, can be thought of as a complex routine, which can be subdivided in different operations: shift, combine, and invoke-rule (left-corner prediction). In order for the parser to merge an input word, this input word must be shifted first, i.e. the parser must take the next input word and check its specifications (such as category) in the lexicon, and place an entry with this information on the stack. Once the entry is on the stack, the parser will try to merge it to the rest of the structure by either combining it with another entry or by invoking a grammar rule (a left-corner prediction). If Merge does not work, the parser will try to Move, i.e. invoke-rule (head-prediction) and insert a trace, or in other words a pointer to some constituent that has already been built. As a last resort, Spell-out will be attempted.

```
(54)  Start                          (First definition)
        Input: $w_1, w_2, w_3 \ldots w_n$
        Stack empty
      Loop
        Merge
          if stack is empty  or Spell-out fails
              then Shift next input word: look word up in the lexicon and
                  create entry <Category, Needed>
      else Combine the entry of category XP on top of the stack with the
            one below it if this needs a constituent of type XP
      else Invoke-rule (left-corner prediction) if the category of the
             entry on top of the stack is XP and there is a rule like
             YP → XP Y ZP
  else Move
      Invoke-rule (head-prediction) if the category of entry on top of the
        stack is X, there is a rule like XP → YP  X  ZP
  else Spell-out either if called by Invoke-rule or as a process of
           reanalysis
end: when (i)     no more input words,
                 (ii)     no incomplete chains
        if there is only one complete entry on the stack:<XP, nil>
              then succeed!
                Else Fail!
```

6.2.5 Some Examples

Taking the algorithm above, and the operations defined in the previous section, we are going to trace some examples step by step. We look at one affirmative sentence, one example of subject movement and one of object movement. We will see when discussing these and some ellipsis examples in the next section that some more instructions must be included in the algorithm defined above. I offer a revised version of the algorithm in (54) with these new operations.

Consider sentence (55) first, and the operations the parser performs for this sentence in (56):

(55) John ate some cookies.

(56) (i) <D, nil> Shift: John

(ii) <DP, nil> Invoke-rule: DP→ D

(iii) <IP, {I, VP}> Invoke-rule: IP → DP I VP

(iv) <V, nil> Shift: ate

<IP, {I, VP}>

At this point, we have two entries in the stack: <V, nil> and <IP, {I, VP}>. We have recognized a verb bottom-up, and we have built an IP node and predicted top-down the I and the VP nodes. These entries as they are cannot be combined; a left-corner prediction is not a possibility either. A head-prediction with rule VP→ DP V DP will be possible if the inflection node is filled (recognized bottom-up). However, since in English *I* can be lexically filled or not, and a verb has already been recognized bottom-up, the parser can safely fill the *I* node with the tense features from the verb. And this is what it is going to do every time that configuration is met, unless the auxiliary is realized up in the tree, to be precise, in node C (we see this with example (61)).

(v)	<V, nil>	Insert tense features for node I
	<IP, VP}	
(vi)	<V, nil>	Invoke-rule: VP→ DP V DP
	<IP, {V, DP}>	
(vii)	<IP, DP>	Combine
(viii)	<D, nil>	Shift: some
	<IP, DP>	
(ix)	<DP, NP>	Invoke-rule: DP→ D NP
	<IP, DP>	
(x)	<IP, NP>	Combine
(xi)	<N, nil>	Shift: cookies
	<IP, NP>	
(xii)	<NP, nil>	Invoke-rule: NP → N
	<IP, NP>	
(xiii)	<IP, nil>	Combine

Now, consider sentence (57) an example of wh-movement. The constituent *who* needs to be related to its base position, as the object of the verb *see*. Let us see how the parser proposed above can handle this construction:

(57) Who$_i$ did you see t$_i$?

(58)	(i)	<D, nil>	Shift: Who
	(ii)	<DP, nil>	Invoke-rule: DP → D

When a CP node is postulated, i.e. when the wh-element is merged as the DP of a rule like CP→ DP C IP, the wh-feature of the wh-element is checked, and a chain is stored. The chain only is not yet complete.

(iii)	<CP, {C, IP}>	Invoke-rule: CP→ DP C IP

Auxiliaries in English can occupy the node I, or the node C (in questions). Whenever the parser creates an entry for an auxiliary it is going to create two different entries: <I, nil> and <C, nil>:

(iv)	<C, nil>	Shift: did
	<CP, {C, IP}>	
(v)	<CP, IP>	Combine
(vi)	<D, nil>	Shift: you
	<CP, IP>	
(vii)	<DP, nil>	Invoke-rule: DP → D
	<CP, IP>	
(vii)	<IP, {I, VP}>	Invoke-rule: IP → DP I VP
	<CP, IP>	

(viii)	<CP, {I, VP}>	Combine
(ix)	<V, nil>	Shift: see
	<CP, {I, VP}>	

The configuration we have here is once again <V, nil> and <CP, {I, VP}>. It was said before that anytime the parser finds this configuration it can fill in the I node with the tense features from the verb that it has just recognized bottom-up, unless there is an auxiliary in C: that is the case here; the auxiliary was merged originally in C, since the sentence is a question. Thus, in this case, the auxiliary in C is related to this position:

(x)	<V, nil>	
	<CP, VP>	
(xi)	<V, nil>	Invoke-rule: VP → DP V DP
	<CP, {V, DP}>	
(xii)	<CP, DP>	Combine

Next step, the wh-element *who* is related to the object position. There is still a chain with that wh-element which is incomplete. This wh-element has not received either case, or a theta-role. In this lower position, this element checks its case and receives a theta-role.

(xiii) <CP, nil> Move-object

I include the definition of this operation in (59) , as well as a graphical representation of what this operation does in (60):

(59) If we have a configuration of the following sort on the stack

<XP, **YP**>

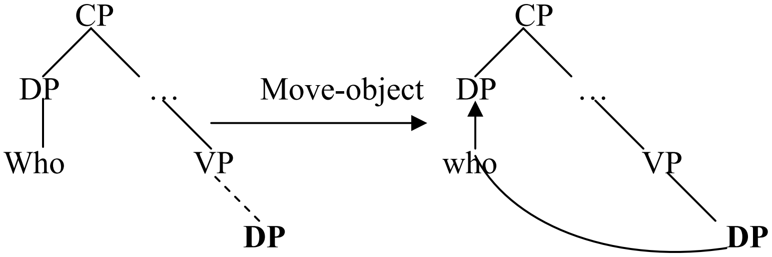
and an incomplete chain with a constituent of type **YP**

then,

YP is assigned a pointer to the constituent that forms the chain, the chain is completed, and the stack is updated in the following manner:

<XP, nil>

(60)



Consider now sentence (61) , an example of subject movement, and the parser operations in (62).

(61) Who t came?

(62)	(i)	<D, nil>	Shift: who
	(ii)	<DP, nil>	Invoke-rule: DP → D
	(iii)	<CP, {C, IP}>	Invoke-rule: CP → DP C IP
	(iv)	<V, nil> <CP, {C, IP}>	Shift: came

As with the node I, C can also be lexically filled or not. After shifting the verb to the stack (step (iv) above), the configuration we have on the stack is <V, nil> and <CP, {C, IP}>. Since C can be lexically filled or not, it is safe to drop an empty category for that position, and keep on building structure.

(v)	<V, nil> <CP, IP>	Insert [e] for node C
(vi)	<V, nil> <CP, {I, VP}>	Invoke-rule: IP → DP I VP
(vii)	<V, nil> <CP, VP>	Insert tense features for node I
(viii)	<V, nil> <CP, {V}>	Invoke-rule: VP→ DP V
(ix)	<CP, nil>	Combine

At this point the parser stops: there are no more input items, there is one complete entry in the stack, and the wh-chain is also complete.

6.2.6 Coordination and Verbal Ellipsis

6.2.6.1 Coordinate Structures

In order to deal with coordinate structures I am going to add rule (63) to our grammar. This rule is augmented with the condition that the elements that are coordinated are alike (this condition has been stated in the linguistics literature as a constraint named Coordination of Likes (Williams 1981)):

(63) BP → XP B YP where XP = YP

I would like to make a comment on the terminology used. BP stands for Boolean Phrase, which is the projection of the Boolean head (B) or coordinator (Munn (1987)). XP and YP stand for any phrasal category. The formulation of this rule, thus, abstracts away from category types and presents a general rule, rather than one rule for every category that could be coordinated.

For a sentence like (64), the stack looks like ((66) step (i)) before the coordinator is shifted; where there is an IP that has been built (the verb's subject *Mary* and object *John* positions have been filled). There is no constituent that is needed for the IP to be complete. The syntactic structure built so far, and kept in the constituent store is that one in (65) After the coordinator is shifted we encounter the situation in ((66) (ii)):

(64) Mary will visit John and Susan.

(65) IP {#DP$_i$:Mary# I:will VP {DP$_i$ V:visit DP {D:John}}}

(66) (i) <IP, nil>

(ii) <B, nil>
<IP, nil>

Now, we cannot combine these two entries in the stack. There is a rule that can be invoked, though. A head prediction can be made with the rule for coordination that was introduced in (63) (BP→ XP B YP). But, in order to do so, there must be some XP that the coordinator is attached to. If we check our Chain Store, we will see that there is no constituent that needs to be moved (no Move operation is allowed). Therefore, the only option is to try Spell-out, i.e. see whether we can attach it to any constituent in our constituent structure in (65).

The lowest XP (see definition of search in (43)) available   in the structure that has been computed is chosen, and the coordinator is attached to that constituent—in this case, the search function will look for a constituent XP, that is, a constituent of any category. The lowest constituent in (65) is the DP *John*; that one is going to be our XP. In order to attach the coordinator to the DP (i) the DP needs to be spelled-out (see (67) step (i)), and (ii) the coordinator is combined with the rest of the structure (see (67) step (ii)). After these operations apply the constituent structure is that one in (68). I represent graphically what the Spell-out operation does in (70):

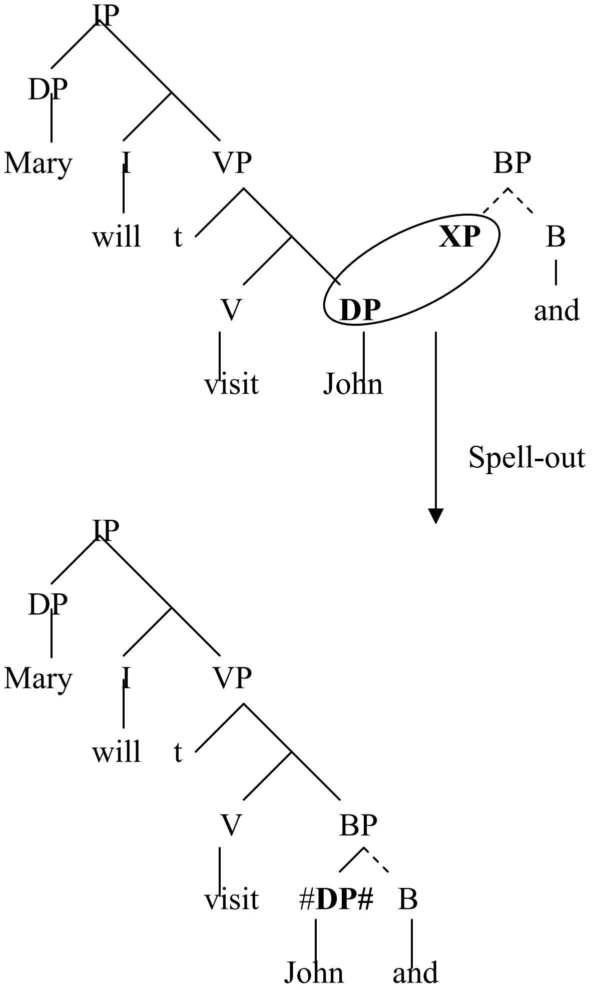
(67) (i) <B, nil> Spell-out DP: BP→ XP B YP

<IP, {B, YP}>

(ii) <IP, YP> Combine

(68) IP {#DP$_i$:Mary# I:will VP {DP$_i$ V:visit BP {#DP:John# B:and}}}

(69)



The next steps are to shift the word *Susan*, invoke the rule DP → D, and Combine this entry with the one below it, giving as a result the configuration in ((70) step (iii)). The constituent structure looks like that in (71) after all these operations apply:

(70) (i) <D, nil> Shift: Susan
<IP, YP>
(ii) <DP, nil> Invoke-rule: DP → D
<IP, YP>
(iii) <IP, nil> Combine

(71) IP {#DP_i:Mary# I:will VP {DP_i V:visit BP {#**DP**:John# B:and **DP**:Susan}}}

The two categories that have been coordinated respect the Constraint of Likes, and the coordination is a well-formed syntactic object.

In the previous example, the initial low attachment of the coordinator is correct; that first attachment does not need to be revised later on. This is not always the case. There are examples where initial low attachment is forced to be analyzed by later incoming input items. Consider sentence (72) now:

(72) Mary will visit John and Susan will call Peter.

For sentence (72), the initial low attachment of the coordinator and the DP Susan will need to be revised as soon as the auxiliary *will* from the second conjunct is merged into the structure. Let us see this in detail. I illustrate the parser operations from the point where the coordinator is shifted to the stack in (73) step (i), next Spell-out applies for the DP *John* (step (ii)), and the coordinator is attached (step (iii)). Next input item to be shifted is *Susan* (step (iv)); the rule DP→ D is invoked then (step (v)), and the resulting DP is combined with the rest of the structure, as the second conjunct (step (vi)). The two categories that the coordinator is combining are of the same type so the constraint on the coordination of likes is respected. At this point, the constituent store has the information in (74) below:

(73)	(i)	<B, nil>	Shift: and
		<IP, nil>	
	(iii)	<B, nil>	Spell-out DP: BP → XP B YP
		<IP, {B, YP}>	
	(iv)	<IP, YP>	Combine
	(v)	<D, nil>	Shift: Susan
		<IP, YP>	
	(vi)	<DP, nil>	Invoke-rule: DP → D
		<IP, YP>	
	(vii)	<IP, nil>	Combine

(74) IP{#DP$_i$:Mary# I:will VP{ DP$_i$ V:visit BP{ #**DP**:John# B-and **DP**:Susan}}}

Next, the auxiliary *will* is shifted (step (viii)), Spell-out applies to the DP *Susan* (step (ix)), and the auxiliary is attached (step (x)). However, attaching the auxiliary to the tree violates the constraint on coordination of likes, since the DP *Susan* is reanalyzed as IP. The syntactic structure computed at this point is that one in (75):

(viii)	<I, nil>	Shift: will
	<IP, nil>	
(ix)	<I, nil>	Spell-out DP: IP→ DP I VP
	<IP, {I, VP}>	
(x)	<IP, VP>	Combine

(75) IP$_1${#DP$_i$:Mary# I:will VP{ DP$_i$ V:visit BP{ #**DP**:John# B-and **IP$_2$** {#DP:Susan# I:will}}}

ERROR SIGNAL!!!

This signals that the constraint on identity of likes has been violated—a DP and an IP are coordinated now—and it enters into reanalysis mode. The coordination needs to be reanalyzed as coordination of two IPs—low attachment needs to be reanalyzed as high attachment. To do so, the constituent store is modified as in (76) into two different command units: (i) IP$_1$ and (ii) the BP where there is an open position that is going to be filled by the first conjunct, once an IP is found:

(76) IP$_1${#DP$_i$:Mary# I:will VP{ DP$_i$ V:visit #**DP:**John#}}
BP{ __ B-and **IP$_2$** {#DP:Susan# I-will}}

Since the coordinator needs to be attached to an IP now, an IP is searched for by spelling-out in a phrase-by-phrase manner. The VP is spelled-out first (represented with the circle and the arrow)—the DP has already been spelled-out—and then the IP is found, and spelled-out. This is the node that the coordinator is attached to. Now, the constituent store can be modified as in (77), the IP$_1$ is assigned to the open position in the BP:

(77) IP$_1${ #DP$_i$:Mary# I:will VP{ DP$_i$ V:visit #**DP:**John#}}
BP{ __ B:and **IP$_2$** {#DP:Susan# I:will}}

(78) #IP$_1${ DP$_i$:Mary I:will VP{ DP$_i$ V:visit **DP:**John}}#
BP{ #IP$_1$# B-and **IP$_2$** {#DP:Susan# I-will}}

Once the coordination has been reanalyzed, the parser keeps on by shifting the verb *call* to the stack, and building the rest of the syntactic structure for the second conjunct:

(79) (i) <V, nil> Shift: call

<IP, VP>

(ii) <V, nil> Invoke-rule: VP→ DP V DP

<IP, {V, DP}>

(iii) <IP, DP> Combine

(iv) <D, nil> Shift: Peter

<IP, VP>

(v) <DP, nil> Invoke-rule: DP→ D

<IP, VP>

(vi) <IP, nil> Combine

Pullman (1986) proposes a lookahead of one word in order to avoid false starts in coordination cases. I do not believe that lookahead of one word solves the problem of where to attach coordinators. In example (72) above, for example, lookahead of one word does not help the parser in attaching the coordinator. It is not until the auxiliary (second word after coordinator) is reached that disambiguation of the attachment site occurs. Increasing lookahead to two or three words does not make the right predictions either, since the disambiguating material (in this case the auxiliary or verb) could come much later. See (80), where the lookahead window would have to be increased to eight words.

(80) Mary will visit John, and that friend of Mary’s who is from Russia will visit Peter.

6.2.6.2 Verb Phrase Ellipsis and Pseudogapping

We now look at two examples of verbal ellipsis: one example of VPE in (81), and one of pseudogapping in (82). In both cases, the coordinator and the second conjunct subject will be first attached low (as the coordination of objects). This low attachment will be reanalyzed as high attachment when the auxiliary *did* is merged to the structure (like in the case of sentence (72) above). We are not going to look at this reanalysis process here, since we already discussed it for example (72). We would rather concentrate on how the gap is detected in both cases:

(81) Peter saw him, and Tom did too.

(82) John talked to Mary, and Peter did to Susan.

Consider first the parser operations for the VPE example, included below. First, the structure of the first conjunct is computed:

(83) (i) <D, nil> Shift: Peter

(ii) <DP, nil> Invoke-rule: DP→ D

(iii) <IP, {I, VP}> Invoke-rule: IP→ DP I VP

(iv)	<V, nil>	Shift: saw
	<IP, {I, VP}>	
(v)	<V, nil>	Insert tense features for node I
	<IP, VP>	
(vi)	<V, nil>	Invoke-rule: VP→ DP V DP
	<IP, {V, DP}>	
(vii)	<IP, DP>	Combine
(viii)	<D, nil>	Shift: him
	<IP, DP>	
(ix)	<DP, nil>	Invoke-rule: DP → D
	<IP, DP>	
(x)	<IP, nil>	Combine

After the first conjunct has been built, the coordinator is shifted and attached low initially (as in (74) above). The DP *Tom* is also attached low:

(xi)	<B, nil>	Shift: and
	<IP, nil>	
(xiii)	<B, nil>	Spell-out DP: BP→ XP B YP
	<IP, {B, YP}>	
(xiv)	<IP, DP>	Combine

(xv) <D, nil> Shift: Tom

<IP, YP>

(xvi) <DP, nil> Invoke-rule: DP → D

<IP, YP>

(xvii) <IP, nil> Combine

Next, the auxiliary is shifted to the stack and merged to the existing structure. In order to attach the auxiliary, what was the coordination of two DPs becomes the coordination of a DP and an IP (see (75)). Since, this violates the constraint on coordination of likes, the parser enters into reanalysis mode (see (76), (77), and (78)):

(xviii) <I, nil> Shift: did

<IP, nil>

(xix) <I, nil> Spell-out DP: IP→ DP I VP

<IP, {I, VP}>

(xx) <IP, VP>

After the low attachment is reanalyzed as high attachment, the parser continues to work with the elements in the stack (the parser stack configuration at this point is briefly discussed and represented in (84) below). There is one entry on the stack "<IP, VP>": there is an IP that needs a VP, but no more input words. However, the top-down prediction of a VP does not need to be confirmed by the recognition of a verb bottom-up, since a functional category like I selects a lexical category like V. Thus, the parser can safely drop a V category in the stack (this operation is defined in (87) below), even if there is no lexical item to support that, and the parser can keep on operating:[12]

(xxi) <IP, VP>

(xxii) <V, nil> Predict VP

<IP, VP>

Once a V category has been dropped, the parser can make a head-prediction, and relate the subject to its base position inside the VP, completing the chain formed by the subject:

(xxiii) <V, nil> Invoke-rule: VP→ DP V

<IP, V>

(xxiii) <IP, nil> Combine

[12] Although this prediction operation is defined here as category specific—only VPs are predicted from IPs—we believe that verb phrases is not the only category that can be predicted. This operation could apply for other categories as well. For example, in Sluicing cases (see (ii) below), a node IP can be predicted if a CP node has been recognized bottom-up:

(ii) I know Mary arrived home, but I don’t know when.

We leave the investigation of which other categories the parse can predict for future work.

In VPE cases, at the point when the VP is predicted, the antecedent has already been spelled-out (this is what *#IP$_1$#* below indicates); in other words, its structure cannot be accessed to interpret the gap. Nevertheless, this is not a problem since a VP prediction can be made without the antecedent's help. Consequently, as we saw in the previous chapter, the antecedent and the gap do not need to be local. The same reasoning applies to pseudogapping cases.

(84) #IP$_1${ DP$_i$:Peter I:e VP{ DP$_i$ V:saw DP:him}}#
BP{ #IP$_1$# B:and IP$_2${ #DP:Tom# I:did}}

Let us consider how the parser operates with the pseudogapping example (82), repeated here:

(85) John talked to Mary, and Peter did to Susan.

Our discussion begins at the point where the gap is encountered. The entry on the stack, at that point, is like in the case of VPE: "<IP, VP>". In this case, there are more input items, but the next input item is not a V. Applying the same reasoning as in the example above, a VP can be predicted without the bottom-up confirmation. A V category is dropped into the stack (see step (ii) below), and the parser continues operating, completing the chain of the subject by relating it to the lower position inside the VP (see step (iii)):

(86)	(i)	<IP, VP>	
	(ii)	<V, nil>	Predict VP
		<IP, VP>	
	(iii)	<V, nil>	Invoke-rule: VP → DP V PP
		<IP, {V, PP}>	
	(iv)	<IP, PP>	Combine

Next, the remnant of the elision is shifted to the stack and combined with the rest of the structure. This is possible, since a VP has already been predicted and built:

(v)	<P, nil>	Shift: to
	<IP, PP>	
(vi)	<PP, NP>	Invoke-rule: PP→ P NP
	<IP, PP>	
(vii)	<IP, NP>	Combine
(viii)	<N, nil>	Shift: Susan
	<IP, NP>	
(ix)	<NP, nil>	Invoke-rule: NP → N
	<IP, NP>	
(x)	<IP, nil>	

In a nutshell, we have seen that for VPE and for pseudogapping examples, once an IP node has been built, a VP node can be built without bottom-up lexical confirmation. I have called this operation Predict-VP, and I include its formal definition below:

(87) If the stack is formed by an entry like <IP, VP>, and

(i) there are no more input items left, or

(ii) the next input item is not of category V

then update the stack in the following manner

<V, nil>

<IP, VP>

The possibility of predicting a VP, once an auxiliary has been recognized bottom-up, is what makes VPE and Pseudogapping examples not subject to any locality restriction. It does not matter whether the antecedent has been spelled-out (#Antecedent#) or not, since it is not accessed to assign structure to the gap.

The case with gapping, however, is different. In gapping examples, the antecedent is vital for the interpretation of the gap. Predict-VP is not a possibility, since there is no overt auxiliary; thus, the antecedent has to be accessed to assign structure to the gap. In sentence (88), the antecedent is available when the gap is reached, since both the coordinator and the DP *Peter* are initially attached low. In (88), the antecedent has already been spelled-out by the time the gap is reached—the initial low attachment of the coordinator and the DP is revised as high attachment when the embedding verb *think* is parsed by reanalyzing the coordination of two objects as the coordination of two IPs, which forces Spell-out of the antecedent VP.

(88) a. John loves Mary and Peter Susan.

b. *John loves Mary and I think Peter Susan.

6.2.7 Algorithm (Revisited)

I add the operations that have been defined in the previous two sections to those that were already included in the first definition of the algorithm:

(89)
```
Loop
  Merge
    if stack is empty  or Spell-out fails
       then Shift next input word: look word up in the lexicon and
            create entry <Category, Needed>
    else Combine the entry of category XP on top of the stack with the
         one below it if this needs a constituent of type XP
    else Invoke-rule (left-corner prediction) if the category of the
         entry on top of the stack is XP and there is a rule like
         YP → XP Y ZP
  else Move
    Invoke-rule (head-prediction) if the category of entry on top of the
```

```
          stack is X, there is a rule like XP → YP  X  ZP
      else Move-object if the entry on top of the stack is <XP, YP>, and
          there is an incomplete chain of category YP.
      else Predict-VP if the entry on top of the stack is <IP, VP>, and
            there are no more input items left or the next input item is
            not a V
    else Spell-out either if called by Invoke-rule or as a process of
        reanalysis
end: when (i)     no more input words,
          (ii)    no incomplete chains, and
      if there is only one complete entry on the stack: <XP, nil>
        then Succeed!
          else Fail!
```

6.3 Conclusion

I have defined a parser and a parsing algorithm based on a minimalist framework, and on the minimalist operations (Merge, Move, and Spell-out). Thus, I have shown that it is possible to think of minimalist grammars as implementable models of language. In order to do so, I take Weinberg's (1999) human processing model, and translate it into a computational model of the sort described by Pullman (1986).

I take Pullman's parser as a starting point and modify its operations, so that they can account for (i) incremental structure building (combining top-down and bottom-up predictions, and attaching nodes basically as they are introduced), (ii) displaced constituents (in examples of syntactic movement), and (iii) coordination and verbal ellipsis. We have seen the algorithm defined above operating with different examples like questions, coordination, and verbal ellipsis. We have also shown that the algorithm can account for all those cases.

CHAPTER 7: CONCLUSIONS

I have offered an account for verbal ellipsis based on the Minimalist Program (Chomsky 1995) framework. The approach developed here is of a syntactic type, i.e. the elision site is considered to be fully structured. I have analyzed ellipsis from a competence, a processing and a computational perspective.

On the competence side, I have shown that ellipsis takes place under identity of syntactic categories. I have argued that by assuming that lexical insertion is a late process in the derivation, ellipsis can be analyzed as a null lexicalization process, and that sloppy and strict readings can be accounted for as the result of interpreting the elided VP at different stages in the derivation—i.e. before of after lexical items have been inserted respectively. Another consequence of having lexical items inserted late is that we can explain those cases of syntactic partial identity (i.e. where there are verbal and agreement morphology differences between antecedent and gap) that seem to put the identity condition on ellipsis under question. In all those cases, identity is respected, but a more abstract notion of identity is needed. I suggest that identity of syntactic categories is the right notion, and that this condition is met before the lexical items are part of the derivation. Thus, we can conclude that the identity condition on ellipsis should be maintained, but also slightly modified—the relevant restriction is identity of syntactic categories—so that it also covers in a unified manner those examples which have been argued to be problematic.

On the performance side, I have accounted for the different locality restrictions in elliptical structures. I have offered a unified syntactic approach which takes into account economy and efficiency considerations, and which makes use of local context. We have seen that in the case of VPE and Pseudogapping, since there is an auxiliary overtly realized, an IP is built, from which a VP can be predicted without the need to resort to the antecedent. On the contrary, in the case of gapping, no structure can be computed without accessing the antecedent, because there is no overt auxiliary or verb that will signal the parser to predict a VP node for the gap. Due to this difference VPE and Pseudogapping are not subject to locality restrictions, but gapping is. I have also accounted for the availability of the antecedent in terms of low initial attachment and spell-out operations which render the syntactic structure unavailable.

As opposed to Lappin &McCord (1990), who propose VP anaphora resolution is done at the level of s-structure, I have defined ellipsis resolution as a two-fold mechanism; some work is done on-line—the gap is detected and a VP predicted from an IP on-line (the case of VPE and Pseudogapping)). Remnants of elision are attached at this point (the case of Pseudogapping)—but there are also certain operations taking place at LF—strict and sloppy readings and quantifier raising operations.

On the computational side, I have shown that it is possible to define an algorithm based on minimalist grammars and minimalist operations. I have translated Weinberg's (1999) algorithm for human sentence processing into a computational model, by taking as a departing point the non-deterministic parser that Pullman

(1986) defines, and modifying its operations in order to account for incremental structure building, verbal ellipsis and coordination.

In a nutshell, I have offered an answer to the identity question from the competence side, but a processing answer to the locality restrictions for verbal ellipsis. I want to suggest that a multidisciplinary approach fares better in the case of ellipsis, and that one should keep an open mind with respect to where answers should be found. In the case of ellipsis, the identity issue can be explained from the competence side of the theory, while the locality issue receives a natural explanation from the processing side—this last one, an issue that has not yet received a good answer from the competence side of the theory.

BIBLIOGRAPHY

Abney, S., and M. Johnson. (1991). Memory Requirements and Local Ambiguities for Parsing Strategies. *Journal of Psycholinguistic Research* 20, 233-250.

Aho, A., and J. Ullman. (1972). *The Theory of Parsing, translation and Compliling*. Prentice-Hall, Inc. Englewood Cliffs, NJ.

Baker, M. (1997). Thematic Roles and Syntactic Structure. In *Elements of Grammar*, ed. Haegeman, L. Kluwer Academic Publishers.

Berwick, B., and A. Weinberg. (1984). *The Grammatical Basis of Linguistic Performance*. Cambridge, Mass.: MIT Press.

---, (1985). Deterministic Parsing and Linguistic Explanation. *Language and Cognitive Processes* 2, 109-134.

Carlson, Katy. (2001). Parallelism and Prosody in the Processing of Ellipsis Sentences. Doctoral Dissertation. University of Massachusetts, Amherst.

Chao, W. (1987). *On Ellipsis*. Doctoral Dissertation, University of Massachusetts, Amherst.

Chomsky, N. (1993). A Minimalist program for Linguistic Theory. In *The View form Building 20: Essays in Linguitics in Honor of Sylvain Bromberger*, eds. Kenneth Hale and Samuel Jay Keyser. Cambridge, Mass.: MIT Press.

---, (1995). Categories and transformations. In *The Minimalist Program.* Cambridge, Mass.: MIT Press.

---, (2000). Minimalist Inquiries: The Framework. In *Step by Step: Essays on Minimalist Syntax in Honor of Howard Lasnik*, eds. Roger Martin, David Michaels, and Juan Uriagereka. Cambridge, Mass.: MIT Press.

Drury, John. (1998). The promise of derivations: atomic merge, and multiple spell-out. *Groninger Arbiten zur Germanistischen Linguistik*, 42, 61-108.

Epstein, S. (1999). Un-Principled Syntax. In *Working Minimalism*, eds. Samuel Epstein, and Norbert Horstein. MIT Press, Cambridge, Mass.

Ferreira, F., and C. Clifton. (1986). The Independence of Syntactic Processing. *Journal of Memory and Language*, 25, 348-368.

Fiengo, Robert, and Robert C. May. (1994). *Indices and Identity*. Cambridge, Mass.: MIT Press.

Fodor, J. D. (1985). Deterministic Parsing and Subjacency. *Language and Cognitive Processes* 1, 3-42.

Fox, D. (1998). Locality in Variable Binding. In *Is the Best Good Enough? Optimality and Competition in Syntax*, eds. Pilar Barbosa, Danny Fox, Paul Hagstrom, Marthe McGiggins, and David Pesestsky. Cambrigde, Mass.: MIT Press.

---, (1995). Economy and Scope. *Natural Language and Semantics* 3, 283-341.

---. (2002). Economy and Semantic Interpretation. *Linguistic Inquiry Monograph* 35. Cambridge MA: MIT press.

Frazier, L., and C. Clifton. (2001). Parsing Coordinates and Ellipsis: Copy α. *Syntax* 4, 1-22.

---, (2000). On Bound Variable Interpretations: The LF-Only Hypothesis. *Journal of Psycholinguistics Research*, 29:2, 125-139.

Frazier, L., A. Munn, and C. Clifton. (2000). Parsing Coordinate Structures. *Journal of Psycholinguistics Research*, 29: 4, 343-370.

Frazier, L., and K. Rayner. (1982). Making and Correcting Errors during Sentence Comprehension: Eye Movements in the Analysis of Structurally Ambiguous Sentences. *Cognitive Psychology* 14, 178-210.

Gibson, E. (1991), Computational theory of Human Language Processing: Memory Limitations and Processing Breakdown. Doctoral Dissertation. Carnegie Mellon University, Pittsburgh, PA.

Goodall, G. (1987). *Parallel Structures in Syntax: Coordination, Causatives, and Restructuring*. Cambridge, Mass.: Cambridge University Press.

Guimarães, M. (1999). Deriving prosodic structure from dynamic top-down syntax. Ms., University of Maryland.

Haik, I. (1987). Bound VPs that Need to Be. *Linguistics and Philosophy* 10, 503-530.

Hankamer, J., and I. Sag. (1976). Deep and Surface Anaphora. *Linguistics Inquiry* 7, 391-428.

Hardt, D. (1993). Verb Phrase Ellipsis: Form, Meaning, and Processing. University of Pennsylvania.

Horstein, Norbert. (1994). LF: The Grammar of Logical Form from GB to Minimalism. Ms., University of Maryland.

---. (1999). Movement and Control. *Linguistic Inquiry* 30, 69-96.

Johnson-Laird, P. (1983). *Mental Models*. Harvard University Press.

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

Kitagawa, Y. (1991). Copying Identity. *Natural Language and Linguistic Theory* 9, 497-536.

Kyle, J. (2001). What VP Ellipsis Can Do, and What it Can´t, but not Why. *The HandBook of Contemporary Syntactic Theory*, eds. Mark Baltin and Chris Collins, Blackwell Publishers, 439-479.

---, (1994). Bridging the Gap. Ms., University of Massachusetts, Amherst.

---, (2000). Few Dogs Eat Whiskas or Cats Alpo. Ms., University of Massachusetts, Amherst.

Laka, I. (1990). *Negation in Syntax: On the Nature of Functional Categories and Projections*. Cambridge, Mass.: MIT Press.

Lappin, S., and M. McCord. (1990). Anaphora Resolution in Slot Grammar. International Business Machines Corporation. Research Division. Research Report RC 15743. Yorktown Heights, N.Y.

---, (1984). VP Anaphora, Quantifier Scope, and Logical Form. *Linguistic Analysis*, 13, 273-315.

Larson, R. (1988). On the Double Object Construction. *Linguistic Inquiry*, 19, 335-391.

Lasnik, H. (1995). A note on Pseudogapping. *MIT Working Papers in Linguistics* 27, 143-163.

---, (1995b). Verbal Morphology: Syntactic Structures meets the Minimalist Program. Reprinted in *Minimalist Analysis*. Blackwell, Oxford, 1999.

Lobeck, A. (1987a). *Syntactic Constraints on Ellipsis*. Doctoral Dissertation. University of Washington, Seattle.

Lόpez, L. (1997). VP_Ellipsis in English and Spanish and the Features of Auxiliaries. Ms. University of Missouri.

---, (1994). The Syntactic Licensing of VP Ellipsis: A Comparative Study of Spanish and English. In M. L. Mazzola (ed) *Issues and Theory in Romance Linguistics*: *Selected Papers form the Linguistics Symposium on Romance Languages XXIII*, 333-354. Washington, DC: Georgetown University Press.

Manzini, M., and A. Roussou. (1997). A Minimalist Theory of Control. Ms.,Univ. di Firenze/Univ. College, London, and Univ. of Wales, Bangor.

Marantz, Alec. (1993). A Late Note on Late Lexical Insertion. In *Explorations in Generative Grammar*, eds. Y-S Kim et al, 396-413. Hankuk Publishing Company.

Martins, A.M. (1994). Enclisis, VP-Deletion and the Nature of Sigma. *Probus* 6, 173-206.

McCord, M. (1980). Slot Grammars. *Association of Computational Linguistics*, 6, 31-43.

Merchant, J. (1999). The Syntax of Silence: Sluicing, Islands, and Identity in Ellipsis. Doctoral Dissertation, University of California, Santa Cruz.

Miller, G.A. and N. Chomsky. (1963). Finitary Models of Language Users. *Handbook of Mathematical Psychology* 2, 419-491, eds. R. Luce, R. Bush and E. Galanter.

Munn, A. (1987a). Coordinate Structures and X-bar Theory. *Mc Gill Working Papers in Linguistics* 4-1, 121-140.

---. (1993). Topics in the Syntax and Semantics of Coordinate Structures. Doctoral Dissertation, University of Maryland at College Park.

Murguia, E. (2000). VP Ellipsis as Null Lexicalization: Strict and Sloppy Readings Emerge at Different Stages in the Derivation. In *Maryland Working Papers in Linguistics (UMWPiL)* 9, 172-202, eds. Maximiliano Guimaraes, Luisa Meroni, Cilene Rodriguez, and Itziar San Martin.

---, (1997). ΣP and VP Ellipsis. Ms., University of Maryland, College Park.

Neijt, A. (1981). Gaps and Remnants: Sentence Grammar Aspects of Gapping. *Linguistic Analysis* 8, 69-93.

---, (1980). *Gapping: A Contribution to Sentence Grammar*. Doctoral Dissertation. University of Utrecht, Dordrecht, Holland.

Nunes, J. (1995). The Copy theory of Movement and Linearization of Chains in the Minimalist Program. Doctoral Dissertation, University of Maryland at College Park.

Otero, Carlos. (1998). Head Movement, Cliticization, Precompilation, and Word Insertion. In *Current Issues in Comparative Grammar: Studies in Natural Language & Linguistic Theory* 35, ed. Robert Freidin. Boston: Kluwer Academic Publishers.

Philips, C. (2003). Linear Order and Constituency. *Linguistic Inquiry* 34, 37-90.

---, (1996). Order and Structure. Doctoral Dissertation. MIT, Cambridge, Mass.

Pritchett, B. (1992). *Grammatical Competence and Parsing Reference*. Chicago: University of Chicago Press.

Pullman, S. (1986). Grammars, Parsers, and Memory Limitations. *Language and Cognitive Processes* 1: 3, 197-225.

---, (1985). A Parser that Doesn´t. *Proceesdings of the Second EACL*, 128-135.

Resnik, P. (1987). Strengthening the No Recursion Hypothesis. Ms, Harvard College, Cambridge, Mass.

---. (1992). Left-Corner Parsing and Psychological Plausibility. *Proceedings of Fourteenth International Conference on Computational Linguistics* (COLING '92), Nantes, France.

Sag, Ivan. (1976). *Deletion and Logical Form*. Doctoral Dissertation, MIT, Cambridge, Mass.

Shapiro, L., and A. Hestvik. (1995). On-Line Comprehension of VP-Ellipis: Syntactic Reconstruction and Semantic Influence. *Journal of Psycholinguistic Research* 24, 517-532.

Shieber, S. (1983). Sentence Disambiguation by a Shift-Reduce Parsing Technique. *Proceedings of the Eighth International joint Conference on Artificial Intelligence* (IJCAI-83), 699-703, Karslruhe, West Germany.

Steedman, M. (1990). Gapping as Constituent Coordination. *Lingusictics and Philosophy* 13, 207-263.

Tancredi, C. (1993). Deletion, Deaccenting, and Presuposition. Doctoral Dissertation. MIT, Cambridge, Mass.

Uriagereka, J. (1999). Multiple Spell-out. In *Working Minimalism*, eds. Samuel Epstein and Norbert Horstein. Cambridge, MA: MIT Press.

---. (1999) Rhyme and Reason: An introduction to Minimalist Syntax. Cambridge, MA: MIT Press

Wasow, T. (1972). Anaphoric relations in English. Doctoral Dissertation. MIT, Cambridge, Mass.

Weinberg, A. (1999). "A Minimalist Theory of Human Sentence Processing." In Working Minimalism, ed. Samuel Epstein and Norbert Horstein. Cambridge, Mass.: MIT Press.

Williams, Edwin. (1977). Discourse and Logical Form. *Linguistic Inquiry* 8, 101-139.

---. (1977). On Deep and Surface Anaphora. *Linguistic Inquiry* 8, 692-696.

Zagona, K. (1988a). Proper Government and Antecedentless VPs in English and Spanish. *Natural Language and Linguistic Theory* 6, 95-128.