DESTROY: A NEW DERIVATIONAL METRIC FOR PARSING*

Valérie GAUTIER

The goal of this paper is to propose a new derivational metric for parsing: *Destroy*. The general claim that underlies this proposal and that I will defend here is that local ambiguities can be solved by simply looking at the way the sentence is generated (its derivational history). Following Phillips (96), I assume that the sentence is built incrementally and lineary. This entails that the insertion of an item in the derivation can destroy the preceding constituent to create a new one. By selecting the derivation which destroys the most the structure already assembled, *Destroy* is the first parsing principle which works in a truly derivational way, contrary to traditional metrics (*Right Association* Kimball, 73 and *Minimal Attachment*, Frazier & Fodor 78, for instance). The article shows that a metric for parsing that appeals to the history of the derivation, such as *Destroy*, accounts not only for well-known cases of local ambiguity, but also provides an explanation of the contrast between garden-path sentences and local ambiguities which do not create any processing difficulties (easy reanalysis).

1. Introduction

Recently, a number of authors have claimed that the connection between ambiguity resolution and syntactic theory is maybe closer than traditionally assumed[1]. Some researchers have argued that principles of grammar can be useful to explain problems of ambiguity resolution (Pritchett 1992 for instance). Following Phillips (96), I make the stronger assumption that grammar and parser belong to the same component. More precisely, I adopt Phillips' claim that structure-building in the grammar proceed the same way as structure-building in the parser, that is,

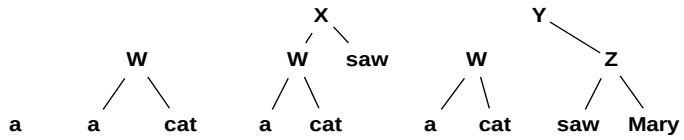
*I would like to thank the following for their helpful comments and suggestions: Hamida Demirdache, Colin Phillips, Juan Uriagereka, Jairo Nunes, Aritz Irurtzun, Nicolas Guilliot.

[1] See Fodor, Bever & Garett, 1974 for classical arguments against this connection.

incrementally and lineary[2]. Under this novel approach, the structure of constituents is evolutive: as the derivation proceeds, the structure previously built is destroyed and rebuilt. The derivation of (1), for instance, proceeds along the lines of (2).

(1) A cat saw Mary.

(2)



Notice that during the course of the derivation, the insertion of *Mary* destroys the constituent previously built [$_X$ a cat] to create the new one [$_Y$ a cat saw Mary]. Phillips argues that a linear derivation resolves both grammar and parsing problems. The fact that constituent-structure is not permanent justifies, for instance, that some constituents can be coordinated but not deleted. Consider (3) and (4) below.

(3) John talked to and gossiped about the kid who sprayed paint on his car.

(4) Helen talked to Jonathan and Alice did___* (to) Matthew.

The way the linear derivation proceeds justifies the contrast between (3) and (4). Building the structure from left-to-right entails that in (3), [talked to] is still a constituent as it has not been lineary merged with a argument. Thus, [talked to] can be coordinated. In contrast, in (4), the addition of *Jonathan* causes the destruction of the constituent [talked to]. As the verb and the prepositon can no longer form a unit, [talked to] cannot be deleted in (4) (for details, see Phillips 2003).

To solve parsing problems, such as local ambiguities, Phillips assumes that *Branch Right* applies.

(5) *Branch Right*

Select the attachment that uses the shortest path(s) from the last input item to the current input item.[3]

As an illustration of (5), consider how the local ambiguity contained in the garden-path sentence (6) is solved by *Branch Right*. In (6), *the sock* can be analysed as

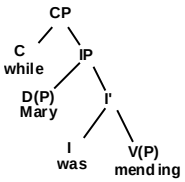
[2] For the classical view that the parser builds structure incrementally and from left-to-right see Kimball 73, Frazier & Fodor 78, for instance.

[3] Reformulated from Phillips (96) p 111.

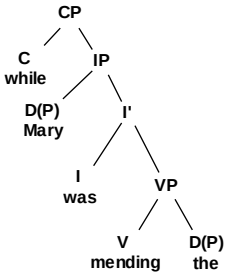
either the object of *mending* or as the subject of the matrix verb. The former parse is choosen: *the sock* must then be reanalysed as a subject

(6) ?While Mary was mending the sock fell off her lap.

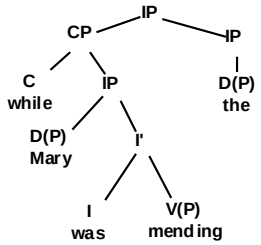
(a)



(b) Object



(c) Matrix subject



To determine which analysis of (6) provides the shortest path, we have to count the number of branches between the heads[4] in the path from V(P) *mending* to D(P)*the*. The attachment in (6)(a) provides the shortest path, (6)(a) is thus selected by *Branch Right*. Note that *Branch Right* compares two representations to select the best attachment. Therefore, Branch Right works representationally[5] like the traditional metrics *Right Association* (Kimball, 73) and *Minimal Attachment* (Frazier & Fodor 78).

Phillips' key proposal is the evolution of constituent-structure (the idea that constituent are built, destroyed and rebuilt). This idea, however, plays a fundamental role in grammar but no role whatsoever in parsing. *Branch Right* is therefore not a principle satisfactory for a unification model (i.e. where grammar & parser belong to

[4]2 branches between V and D(P) in 0(a) vs 6 branches between V(P) and D(P) in 0(b).

[5]It is not completely accurate to say that Phillips' analysis is representational. The counting has indeed sometimes to be made in two steps to reach the preferred parse. Note, however, that in most examples, (including (6)) using *Branch Right* derivationally yields strictly the same result as using it representationally.

the same component). In contrast, I will argue that the fact that constituents are not permanent is not only relevant but crucial to choose between two analyses of a given sentence. Thus, I propose a metric which doesn't compare two structures, but the two ways these two structures have been built.

(7) *Destroy*: choose the derivation that destroys the most the last constituent.

The paper is organized as follows. In section 2, I show how the linear derivation proceeds step by step and how *Destroy* applies. Section 3 shows that *Destroy* accounts for well-known local ambiguities. Finally, Section 4 argues that the history of the formation and destruction of constituents can account for reanalysis, an issue left unexplained by Phillips.

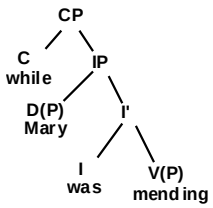
2. Proposal: Destroy, a derivational metric

Let's start by briefly illustrating the proposal. Consider how (6), repeated in (8), is disambiguated according to (7).

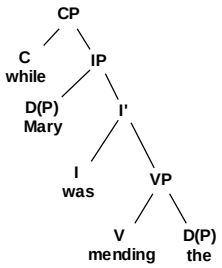
(8) ?While Mary was mending the sock fell off her lap.

First, suppose that the structure in (8)(a) has already been built.

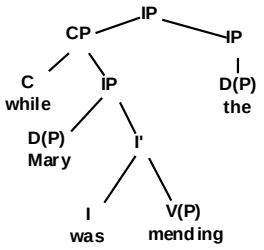
(a)



(b) Object



(c) Matrix Subject



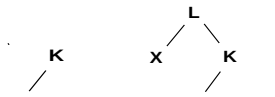
(8)b shows that the insertion of the DP as an object triggers the destruction of I' [was mending]. In contrast, in (8)c, attaching *the* as the matrix subject doesn't cause the destruction of the constituent I' [was mending]. Recall that the preferred analysis is the one that destroys the most the structure previously built. Thus, *Destroy* correctly predicts that object attachment of the DP is preferred to matrix subject attachment. Before going through the empirical coverage of *Destroy*, let us see precisely how to define the operation which destroys a constituent to create a new one.

2.1. Definition of Top-Down Merge

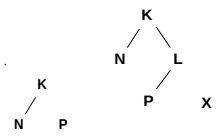
From a minimalist perspective, the derivation of a given sentence is built by the two structure-building operations *Merge* and *Move*[6]. *Merge* is traditionally assumed to apply to the root[7] of the tree, i.e. Merge is cyclic (cf (9)(a)). Notice that, as a linear derivation usually proceeds in a top-down manner, *Merge* crucially doesn't increment the tree at the root: each time a new constituent is inserted in the derivation, it replaces something in the structure, i.e. Merge is counter-cyclic[8] (cf (9)(b)).

(9)

(a) Cyclic Merge
Merge X and K yields the new constituent L. L dominates the tree.



(b) Counter-cyclic Merge
Merge P and X yields the new constituent L. L doesn't dominate the tree, but L *replaces* P in the tree.



Since *Merge*, when it applies top-down, is a counter-cyclic operation, I propose the definition in (10), based on Kitahara 93[9].

(10) *Top*-down *Merge*[10]
  a) Applied to α and β, Merge forms the new object {L, {α, β}}
  b) L replaces the lowest and rightmost term of the structure.

[6]Due to space limitations, I won't consider *Move* in this paper.

[7]"Merge always applies in the simplest possible form: at the root". (Chomsky, 1995)

[8]see footnote (10)

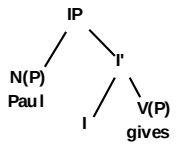
[9]"Applied to two object α and Σ with β, Merge forms Σ' by merging α and β. This operation, if noncyclic, replaces β in Σ by K = {γ, {α, β}}." Kitahara (1993)

Let us illustrate (10) by presenting the relevant steps of the derivation of the unambiguous sentence in (11).

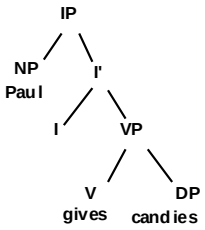
(11) Paul gives candies to Marc.

The first relevant step of the derivation is given in (11)(a).

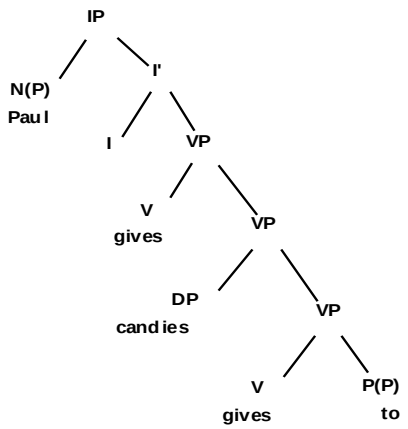
(a)



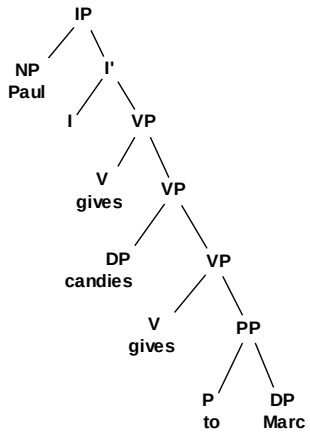
(b): V(P) is merged with the incoming item DP *candies*, forming the constituent {VP{V, DP}. This constituent replaces V(P).



(c): A copy of the verb V*gives* and the P(P)*to* are concatenated, forming the constituent {VP{V, P(P)} replacing DP *candies*



(d): P*to* and DP *Marc* are merged, the new constituent *to Marc* replaces *to*.



[10]Note that in a linear derivation, *Merge* can sometimes increment the structure cyclically. These other instances of *Merge* are presented in section 2.2.

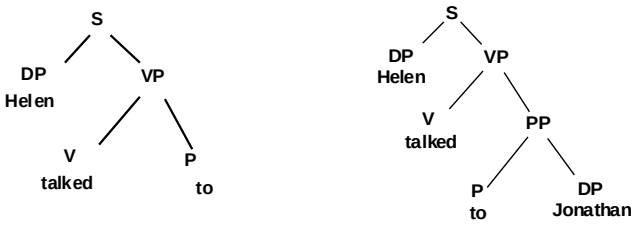
I have just defined how a top-down derivation proceeds. I now turn to the issue of how the parsing metric *Destroy* proposed in (7) applies. To this end, I will define more precisely how *Destroy* works by presenting a scale of destruction for *Merge*.

2.2. Scale of destruction for Merge

A item arriving in a derivation can be inserted by three different structure-building operations: Merge which destroys the last constituent, Merge which adjoins to the last constituent or Merge which doesn't affect the last constituent. I define these three operations below.

Merge which destroys the last constituent[11]: This type of structure-building operation holds when the derivation is incremented by *top-down Merge*, as defined in (10). For instance, the insertion of *Jonathan* in (12)(b) destroyed the last constituent created in (12)(a), [talked to].

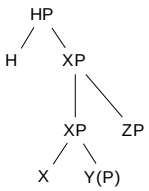
(a) (b)



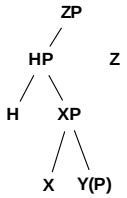
Merge which adjoins to the last constituent: This operation involves adjunction of a term to any term of the last constituent. In (13), ZP has been adjoined to the last constituent XP, creating a mother node bearing the same label as XP. Notice that while *Merge which adjoins* adds a new term to XP, it doesn't destroy it in the same way as *Merge which destroys*. Indeed, the application of the latter operation would have destroyed the unit [X Y(P)] whereas the application of adjunction preserves the integrity of the constituent [X Y(P)].

[11] Note that I've still not defined what should count as the last constituent. I turn to this issue in section 3.4

(13) Merge which adjoins to XP



(14) Merge which doesn't affect XP



Merge which doesn't affect the last constituent: This instance of *Merge* holds when you add an item anywhere but to the last constituent. In (14), ZP has been merged to the constituent HP without affecting the last constituent XP.

We have now a scale of destruction which will allow us to determine how our metric applies.

(15) Metric
Destroy: choose the derivation that destroys the most the last constituent

(16) Scale of destruction for *Merge*
Merge which destroys α > Merge which adjoins to α > Merge which doesn't affect α.

3. Empirical coverage

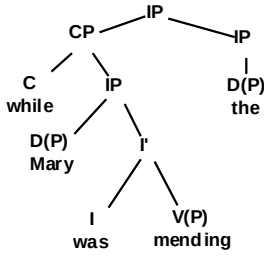
3.1. Merge which destroys α vs Merge which doesn't affect α

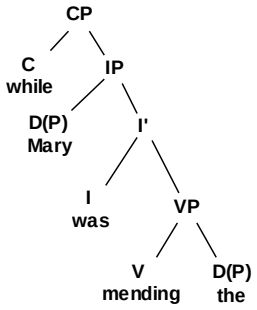
Reconsider our previous example, repeated in (17) below.

(17) ?While Mary was mending the sock fell off her lap.

(a) Object

(b) Matrix subject



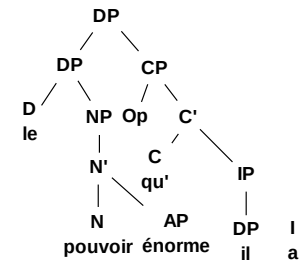


Merge destroys [was mending] *Merge* doesn't affect [was mending]

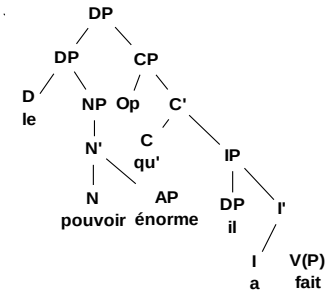
As defined in (16), *Merge which destroys α* is more destructive than *Merge which doesn't affect α*. Thus, *Destroy* correctly predicts the preferred parse for (17) to be (17)(a).

Our new derivational metric will also yield the correct interpretation for the french garden-path sentence in (18), which is another case of competition between *Merge which destroys α* and *Merge which doesn't affect α*. (18) is a garden-path because *fait* can be analysed as either a past participle or an active verb. There is a strong preference for analysing *fait* as a past participle, which then makes the rest of the sentence unparsable. The only option would then be to reanalyze *fait* as the matrix verb.

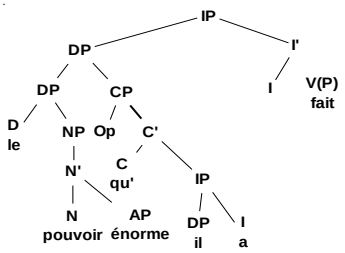
(18) ?Le pouvoir énorme qu'il a fait de lui l'homme le plus respecté de la ville. (Wehrli, 97)
?The huge power that he has made him the most respected man of the city.
The huge power he has, makes him the most respected man of the city.
(a)



(b) Embedded past participle



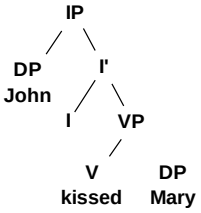
(c) Matrix verb



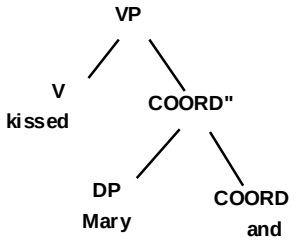
The attachment of *fait* in (18)(b) requires the destruction of the constituent [il a]. In contrast, analysing *fait* as the matrix verb in (18)(c), doesn't destroy this constituent. Since our new metric selects the derivation which destroys the most the structure already built, *Destroy* correctly predicts the past participle attachment to be the preferred one.

(19) is the last example of the competition between Merge *which destroys α* and *Merge which doesn't affect α*. In (19), at the point where we encounter *and*, it can have two readings. *and* can coordinate the two DPs *Mary* and *her sister*, or two clauses *John kissed Mary and her sister laughed*. DP coordination is incorrectly preferred, and (19) will therefore have to be reinterpreted as a coordination of clauses.

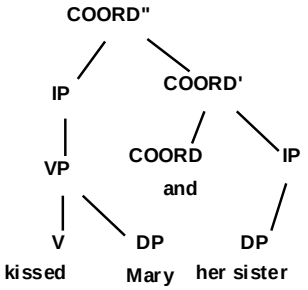
(19) ? John kissed Mary and her sister laughed.
Suppose that (19)(a) has already been built.
(a)



(b) Object coordination



(c) Clausal coordination



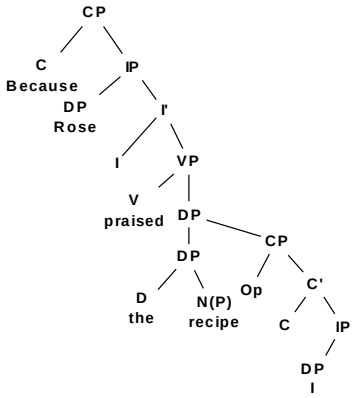
Notice that in (19)(a), the VP [kissed Mary] has been destroyed, while in (19)(b), the application of the operation *Merge which doesn't affect α* has left the VP intact (*and* having been merged with IP). As (19)(a) is the preferred analysis of (19), our derivational metric makes the correct prediction for a garden-path sentence like (19).

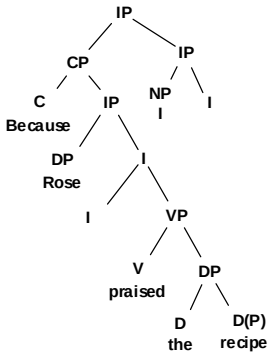
3.2. Merge which adjoins to α vs Merge which doesn't affect α

The local ambiguity in (20) lies in the fact that the DP *I* can be analysed as the subject of a relative adjoined to the DP *the recipe* or as the subject of the matrix verb *made*. There is a strong preference for the former attachment, which leads the parser to a garden-path.

(20) ? Because Rose praised the recipe I made it for her birthday.

(a) Relative clause attachment (b) Matrix attachment





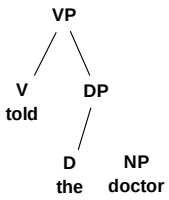
(20)(a) shows that adjunction of the relative clause has destroyed the constituent [praised the recipe]. In contrast, the alternative derivation, presented in (20)(b), doesn't affect the last constituent [praised the recipe]. Recall that *Merge which adjoins to α* is more destructive than *Merge which doesn't affect α*, consequently, *Destroy* correctly chooses the relative clause attachment.

3.3. Merge which destroys α vs Merge which adjoins to α

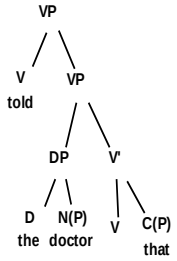
The garden-path in 21 is an example of competition between *Merge which destroys α* and *Merge which adjoins to α*. In 21, the string *that he was having trouble with* can be analysed either as a complement of the verb *tell* or as a relative attached to the DP. There is a strong preference for analysing the embedded clause as the object of *told*. This attachment makes the sentence ungrammatical: the clause *that he was having trouble with* must then be reanalysed as a relative.

(21) ?The man told the doctor that he was having trouble with to leave.

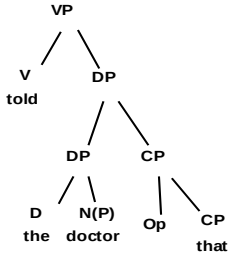
(a)



(b) verb attachment



(b) DP adjunction



In 21(b) *Merge which destroys α* has applied, destroying the VP built in 21(a). In contrast, in 21(b) it's the operation *Merge which adjoins to α* which has applied, adjoining CP to [the doctor]. The preferred analysis is 21(b), as correctly predicted by our metric *Destroy*.

3.4. Defining the last constituent

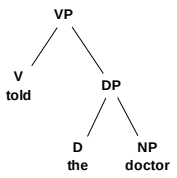
Notice that we have tacitely assumed that the last constituent in 21(a), as well as in (19)(a) and (20)(a), is VP, and not DP. Why? Recall that *Destroy* crucially does not apply representationally but derivationally. I thus assume that *the last constituent is not the last constituent inserted* (DP) in the derivation, but *the last constituent built or re-built/destroyed* (VP) by the derivation. In other words, the last constituent is the last constituent affected by *Top-down Merge*, repeated in (22).

(22) *Top-down Merge*
a) Applied to α and β, Merge forms the new object {L, {α, β}}
b) L replaces the lowest and rightmost term of the structure.

To see that the last constituent (re)built is indeed VP and not DP in 22(a), consider the derivational steps needed to obtain the structure [told the doctor].

(a)

(b)



When *the doctor* replaces *the* in VP, [the doctor] is the new constituent entered in the derivation, but [the doctor] is not the last constituent affected by (22). VP is the last constituent affected by (22). I therefore reformulate *Destroy* as in (23).

(23) *Destroy*: choose the derivation that destroys the most the last constituent built.
Last constituent built = last constituent affected by Top-down Merge.

4. Towards a theory of reanalysis: don't rebuilt what you have destroyed

This final section focuses on an important issue in sentence processing not addressed by Phillips (96): the problem of reanalysis. Certain locally ambiguous structures fail to result in processing difficulty while others lead invariably to garden-path effects. Consider, for example, the constrast between (17) (repeated as (24)), and (25).

(24) ? While Mary was mending the sock fell off her lap.

(25) Steve had known Max hated sharks.

The attachement of the DP *the sock* in (24), just like that of the DP *Max* in (25) is ambiguous between an object and a matrix subject. In the two examples, the preferred attachment is the object one. However, while in (24), it is costly to reanalyse *the sock* as a subject, the reanalysis of *Max* in (25) doesn't cause any processing difficulty. Although *Branch Right* picks out the preferred parse for the two sentences, it doesn't account for the fact that (25) is perfectly processable while (24) is not. I propose an explanation for this contrast below.

My hypothesis is based on a core property of the structure-building operation as defined by Phillips. In particular, Phillips argues that the way the linear derivation proceeds makes the following prediction: "once a constituent has been destroyed, it is no longer available to any syntactic processes[12]" (Phillips 2003). Once again, if we take seriously the claim that grammar and parser belong to the same component, Phillips' hypothesis must be generalized for parsing as in(26):

(26) Once a constituent has been destroyed, it is no longer available to both syntactic and parsing processes.

I take the generalization in (26) to underlie the contrast between costly reanalysis and cost-free reanalysis. Reanalysis will be costly if it must resort to unavailable constituents. That is, constituents destroyed during the first and preferred parse. On the contrary, reanalysis will be easy as long as it doesn't involve unavailable constituents. As an illustration of the proposal, consider the derivations of (24) and (25), repeated in (27) and (28).

[12] See (3) and (4) in this paper.

(27) ?While Mary was mending the sock fell off her lap.	(28) Steve had known Max hated sharks
a) **[was mending]**	a) **[had known]**
b) [was [mending the sock]]	b) [had [known Max]
c) **[was mending]** [the sock fell off her lap]]	c) **[had [known** [Max hated sharks]]]

Notice that the reanalysis of *the sock* entails backtracking: the constituent [was mending] which has been created at step (27)(a) and destroyed at step (27)(b), is re-built at step (27)(c). On the contrary, no backtracking is necessary in (28). That is, step (28)(c) doesn't involve a constituent no longer available: the substitution of the DP *Max* by the CP *Max hated sharks* affects the VP *known* but does not require re-building any constituent. In sum, by looking at the history of the derivation, we not only solve local ambiguities, but also explain when reanalysis is costly or not.

5. Conclusion

I have argued that local ambiguities can be explained by the way the sentence is generated. In particular, I've proposed a new structural ambiguity resolution principle, *Destroy*. The key idea in this proposal is that constituent-structure is not permanent (cf. Phillips 2003). Unlike traditional metrics (*Right Association*, *Minimal Attachment*, *Branch Right*) *Destroy* doesn't compare two representations but the two possible derivations of a given sentence.

First, I demonstrated that *Destroy* covered well-known cases of local ambiguities. I then showed that the history of the formation and destruction of constituents can tell us when reanalysis is costly or not[13]. If this proposal is correct, the history of the derivation is relevant not only for grammar, but also for parsing.

References

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

Fodor J., T. Bever & M. Garett (1974), *The Psychology of Language. An introduction to Psycholinguistics and Generative Grammar*. New York: McGraw-Hill.

Frazier, L. & J. Fodor (1978) 'The sausage machine: a new two-stage parsing model', *Cognition* 6: 291-325.

[13] Notice that our analysis correctly predicts that all cases of competition between *Merge which destroys α* and *Merge which doesn't affect α* causes processing difficulty (see (18) and (19)). Cases of competition relating to adjunction and others instances of competition will be discussed in Gautier forthcoming.

Kimball, J. (1973) 'Seven Principles of Surface Structure Parsing in Natural Language', *Cognition* 2: 15-47.

Kitahara, H. (1993) *Target α: a unified theory of movement and structure-building*, Doctoral Dissertation. Harvard University, Cambridge, Mass.

Phillips, C. (1996) *Order and Structure*, Doctoral Dissertation, MIT.

Phillips, C. (2003) 'Linear order and constituency', *Linguistic Inquiry* 34: 37-90.

Pritchett, B. (1992) *Grammatical Competence and Parsing Performance*, Chicago: The University of Chicago Press.

Wehrli, E. (1997) *L'analyse syntaxique des langues naturelles*, Paris: Masson.

Weinberg, A. (2000) 'A minimalist theory of human sentence processing', in: S. D. Epstein & N. Horstein (eds.), *Working Minimalism*, Cambridge, Mass: MIT Press.