Coordination and constituency

Houtman, Joop

3 A product-based applicative categorial grammar (PACG) for coordination

3.1 Introduction

In chapter 2, we discussed the advantages and disadvantages of a categorial grammar which consists of flexible rules, like Functional Composition and Type Lifting. We showed that there are four arguments against this kind of flexibility that cannot easily be overcome in flexible systems like the Lambek-calculus. The conclusion reached in that chapter was that we must seriously search for categorial grammar systems for Dutch, and in particular Dutch coordination phenomena, that do not include Composition and Lifting rules.

In this chapter, I will present an applicative categorial grammar for Dutch coordination phenomena. The grammar consists basically of application rules, product rules, and combinations of both. The grammar will not include flexibility increasing rule schemata, like Functional Composition and Type Lifting. One exception must be made for the verb clustering phenomena in Dutch, which appears to call for a rule of Disharmonic Composition. In chapter 1, we discussed the possibility of treating verb clustering as a lexical phenomenon. Nevertheless, this doesn't relieve us from the necessity of incorporating a rule for Disharmonic Composition in a categorial syntax.

The grammar differs from the present standard of categorial grammars as logical deduction systems. Logical deduction systems consist of axioms and inference rules. Depending on the exact set of axioms and inference rules, other valid inferences can be deduced. The Lambek-calculus we discussed in chapter one, is such a system. The grammar I propose is a product-based, applicative categorial grammar (PACG) for Dutch coordination phenomena. The majority of the rules of the grammar can be thought of as building up derivation trees. Besides the rules of the grammar, no other inferences can be deduced. Some rules are universal in languages, like the application rules and the product rules, but others are more likely to be language specific rules, like the Coordination Rule, the rule

for Disharmonic Composition, and the Gapping Decomposition rule. This last rule accounts for gapping phenomena in Dutch.

The grammar I propose has its roots in work by Mark Steedman.[37] In some ways the grammar resembles Cremers's *minimal grammar* (Cremers (1993)). In using the product operator, the grammar I propose here is connected to Mary Wood's *Thursday Grammar*, presented in Wood (1988).

In using the PACG, the disadvantages mentioned in section 2.3 with respect to the Lambek-calculus can successfully be avoided. We will discuss this in section 3.2. In Wood (1988) and Cremers (1993) the notion *minimal grammar* is used to refer to a grammar that is 'applicative only'. Because there is an AB-grammar for each Lambek-grammar which has the same weak generative power (cf. Pentus (1992)), Cremers states that we should use the equivalent AB-grammar. In such a way it should be possible to avoid the overgeneration that becomes evident in a Lambek-grammar when we introduce rules for Disharmonic Composition. *Minimal grammar* implies a reductional system with as few rules as possible, and with no structure building rules, like Type Lifting. These rules change the category of an expression, which can result in new derivational possibilities. Minimality refers obviously to the syntactic component of the grammar. Such a grammar will, on the other hand, increase the lexicon. In this sense the notion *minimality* is less adequate. Cremers's thesis was devoted to a parsing procedure which would come to the right conclusions about coordinative structures. His starting point is, however, quite different from the one underlying the proposal in this thesis. In Cremers's view coordination cannot be described or understood in terms of constituent structure or grammatical function. In his view coordination is only a procedural matter. I don't agree with that. Although coordination steps across constituent structure quite easily, we should, nonetheless, try to supply a purely grammatical account for coordination phenomena. The most important argument for this is that grammaticality issues should be decided by the declarative grammar and not by procedural mechanisms of some sort.

---

[37] See for example Ades & Steedman (1982), Steedman (1983), Steedman (1984), Steedman (1985), Steedman (1987) and Steedman (1990).

In section 3.2, I will discuss the product-based applicative categorial grammar (PACG) for Dutch coordination phenomena. Compared to the Lambek-calculus, the PACG is less flexible with respect to the use of Composition rules and Type Lifting rules, less flexible with respect to conjoinability of arbitrary modifiers, and less flexible with respect to constituent structure. On the other hand, some kind of flexibility is called for, even in the PACG. The reason for this is that we have to account for the coordination phenomena. This flexibility is taken care of by the associativity of the product categories. This associativity makes the core part of the grammar, the universal rules, structurally complete. Nevertheless, the flexibility caused by the associativity of product categories is harmless to grammatical function, because it doesn't affect function/argument relations whatsoever.

In section 3.3, I will compare the PACG to Wood's Thursday Grammar. Wood's application of the product rule is somewhat restricted. She uses it, for example, to combine the two objects (X and Y) of a ditransitive verb to form an expression of category X*Y. These combined expressions can coordinate with similar expressions. To complete the derivation, Wood transforms the category of the ditransitive verb (Z/Y)/X to Z/(X*Y), by what she calls *Flattening*. This category yields a Z, when applied to an X*Y. We will see that in some cases Wood needs a rule to take the product apart, the *Deconstruction Rule*. I will show that by combining application rules and product rules, the grammar doesn't need such mechanisms. Of course, in the product rules and the derived product rules, the PACG is richer than Wood's grammar.

Section 3.4 will reopen the discussion on constituency, and I will claim that constituency of words and word sequences depends on whether or not the categories of these expressions contain the product operator. This way, we are able to restore content to the notion *constituent* considerably.

In section 3.5 I provide some extensions and adjustments of the categorial system on the ground of Gapping phenomena, and combined Gapping and Right Node Raising phenomena. In this section I also pay attention to the approach of Morrill and Solias (1993) to Gapping and other phenomena.

In section 3.6 some concluding remarks are presented.

3.2 A product-based applicative categorial grammar (PACG)

3.2.1 Introduction

In this section, 3.2, I will present the product-based applicative categorial grammar (PACG) for Dutch. In section 3.2.2, we will discuss the motives for this PACG. These motives are brought about by the disadvantages of the Lambek-calculus in the account of coordination phenomena. The four arguments against the Lambek-calculus as a descriptive system for Dutch, discussed in chapter 2, will be shown not to hold with respect to the PACG.

The sections 3.2.3 and 3.3.4 discuss the PACG in detail. The first of these sections is about the universal part of the PACG, while the second presents the language specific rules.

In chapter 2, we have seen that restrictions on the Coordination Rule cannot be formulated within a Lambek-system. We can always get round such restrictions by means of Functional Composition and Type Lifting. In section 3.2.5, I will show that restrictions on the coordination rule can be accounted for in the PACG.

In section 3.2.6, we will observe that Across-the-board dependencies can be accounted for straightforwardly within the PACG. In section 3.2.7, some conclusions are drawn.

3.2.2 Motives for a PACG

An important requirement for a grammar of a natural language L is that all sentences of L are recognized as such, without recognizing the other word sequences as sentences. In categorial grammar the most important part of this ‘all-and-only’ principle has always been the ‘all’-part. The central issue in categorial linguistics has mostly been: what means do we need to describe the phenomena (without asking what should be done to prevent over-recognition). This has lead to highly flexible categorial systems, starting from the Lambek-calculi (Lambek (1958),

Lambek (1961)) to still more flexible systems Lambek+Permutation (LP), LP+Contraction (LPC), LP+Expansion (LPE) and LP+Contraction and Expansion (LPCE) presented by Van Benthem (Van Benthem (1989)) and Moortgat (Moortgat (1988)). We discussed these calculi in chapter 1. The emphasis on flexible categorial systems is partly due to coordination phenomena (see Zwarts (1986)). Flexibility is called for, because coordination operates on levels different from the constituent level, as we saw in chapter 2, section 2.2. Most efforts have been made within the Lambek-calculus, a highly flexible system in the sense that each two adjacent expressions form a constituent. The rules for Functional Composition: Right Composition (RC) and Left Composition (LC) and the rules for Type Lifting (TL), introduced in chapter 1, are responsible for this. In chapter 2, we argued that flexibility has a great impact on languages that exhibit disharmonic structures. By adding Disharmonic Composition (1) to the Lambek grammar, we obtain a grammar which is intermediate between Lambek and LP, Lambek+Permutation. This grammar allows for too many permutations to be adequate for the description of natural languages with disharmonic structures.

1. **Disharmonic Composition (DC)**
   a. X/Y Z\Y ⇒ Z\X
   b. Y/Z Y\X ⇒ X/Z

If we want to work on the 'only'-requirement of a categorial grammar for Dutch, we will have to abandon the Lambek-calculus as universal grammar for natural languages. Assuming that Dutch is indeed a disharmonic language (evidence for this we find for example in Moortgat (1988) and in Cremers (1993)), we have to search for categorial systems of which the universal parts are less strong than the Lambek-calculus. If we take into account that for every language recognized by the Lambek-calculus there is an applicative grammar (AB-grammar) that also recognizes that language (Buszkowski (1985), Pentus (1992)), then there is no a priori need to consider the Lambek-calculus as the weakest grammar to describe natural languages.

The grammar I propose, the PACG, is less flexible than the Lambek-calculus in that it doesn't allow for (Harmonic) Functional Composition and Type Lifting. In the Lambek-calculus, these two rules, among many others, are derivable. The combination of a Lambek-calculus and disharmonic structures allows for more

permutation than is warranted by rigid word order languages, like Dutch. The PACG doesn't have this undesirable feature, because Harmonic Composition and Type Lifting are completely banned.

On the other hand, the universal grammar, that is, the grammar consisting of the product rules and the application rules, is by the nature of the product rules structurally complete. The structural completeness is lost, when we add the language specific rules, like the Coordination Rule, to the grammar. The completeness of the universal grammar doesn't lead, however, to the undesirable consequences the usual flexible grammars suffer from. The reason for this is that the only way to reduce function/argument structures is by the rules for application (and the Disharmonic Composition rule). In fact, by adding the product rules to the applicative categorial grammar, the coordination phenomena in Dutch are accounted for, without supplying the rich structures of the Lambek-calculus. Product rules, in fact, postpone reduction.

As is immediately clear from the grammar proposed, Dekker's paradox will not play any role in the PACG. Without Harmonic Functional Composition and Type Lifting, we are not confronted with the proof of two arbitrary modifiers to be lifted to the same category.

The restrictions on the Coordination Rule, which couldn't be maintained within the Lambek-calculus, will be reformulated for the PACG in section 3.2.5.

The notion *constituency* will be discussed, as I mentioned in the introduction 3.1, in section 3.4.

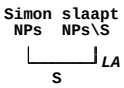
In the next sections, 3.2.3 and 3.2.4, we present the product-based applicative categorial grammar for Dutch. The grammar consists of a universal part and a language specific part.

3.2.3 Universal rules

In this section, I present the universal rules of the PACG, which covers most function/argument relations in natural languages. Together with the language specific rules for Dutch, presented in section 3.2.4, they yield a categorial grammar for Dutch, especially for Dutch coordination phenomena. We see that as for the universal rules, the domain of application is in no way restricted. Where the language specific rules are concerned, domain restrictions come into play.

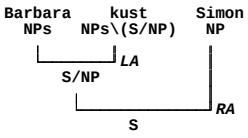
The universal rules include the reduction rules *Right Application* (RA) and *Left Application* (LA). These are the minimal requirements for a directional categorial grammar. The function categories specify the domain of the arguments to be found and the range category of the complex sequence. In example (2) the verb *slaapt* (*sleeps*) is a left oriented function from NP_s to S. The function applies to the argument *Simon*, which is an NP_s occurring to the left of the function category. After functional application the sequence *Simon slaapt* is recognized as a sentence. In example (3) the transitive verb *kust* (*kisses*) is a left oriented function from NP_s to (S/NP). That means the range of the function is itself a function, one from NP to S.

2.



Simon sleeps

3.



Barbara kisses Simon

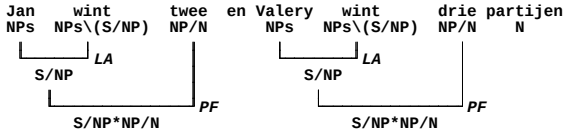
4. **Right Application (RA)**
   X/Y Y ⇒ X

5. **Left Application (LA)**
   Y Y\X ⇒ X

Besides the reduction rules RA and LA we need product rules. The reason for this is that within an applicative grammar we should one way or another deal with non-constituent coordination. Since we do not have access to Type Lifting rules and rules for Functional Composition, we need product rules to conform to the Coordination Rule, stated in section 3.2.4.

The first product rule is *Product Formation* (PF). It states that two adjacent sequences with categories X and Y can form a complex sequence with category X*Y, an ordered pair of categories. Example (6) shows why we need the rule PF (7).

6.



Jan wins two and Valery wins three games

7. **Product Formation (PF)**
X Y $\Rightarrow$ X*Y

The second product rule is the rule for *Product Associativity* (PA). The PA-rule accounts for the non-relevance of the order in which more than two expressions are put together by PF. So, if we have three expressions with categories X, Y and Z, we can put them together by PF from left to right to yield an expression of category (X*Y)*Z. We can, however, also work from right to left. In this case the complex expression receives the category X*(Y*Z). The PA-rule (8) accounts for the equivalence of these two categories.

8. **Product Associativity (PA)**
(X*Y)*Z $\Leftrightarrow$ X*(Y*Z)

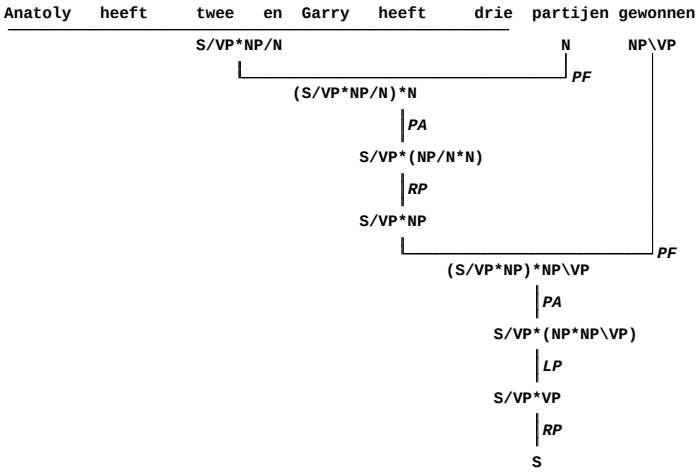
I suppose the rules introducing the product operator to work only from left to right. This means, products can be introduced by PF, but they can only be eliminated by reduction rules. This means, the grammar doesn't include pure rules of product-elimination. The PA-rule does not introduce instances of the product operator. Therefore, it is supposed to work in both directions.

Apart from the Reduction Rules and the Product Rules, we need a third kind of rule, namely the *Derived Reduction Rules*. There are two of these rules, the *Right Product Application* (RP) and the *Left Product Application* (LP). These rules account for the applicability of the reduction rules, even if functor and argument are already tied together by the product operator, and even if these expressions are embedded in larger parts, generated by the product operator. The rules RP (9) and LP (10) are illustrated in example (11).

9. **Right Product Application (RP)**
$\Gamma^*((X/Y)^*Y)^*\Delta \quad \Rightarrow \Gamma^*X^*\Delta$

10. **Left Product Application (LP)**
$\Gamma^*(Y^*(Y\backslash X))^*\Delta \quad \Rightarrow \Gamma^*X^*\Delta$

11.



‘Anatoly has two and Garry has three games won’
Anatoly has won two games and Garry has won three games

I present the universal rules of the PACG in (12).

12. **Universal Rules**

Reduction Rules

Right Application (RA)
X/Y Y $\Rightarrow$ X

Left Application (LA)
Y Y\X $\Rightarrow$ X

Product Rules

Product Formation (PF)
X Y $\Rightarrow$ X*Y

Product Associativity (PA)
(X*Y)*Z $\Leftrightarrow$ X*(Y*Z)

Derived Reduction Rules

Right Product Application (RP)
Γ*((X/Y)*Y)*Δ $\Rightarrow$ Γ*X*Δ

Left Product Application (LP)
Γ*(Y*(Y\X))*Δ $\Rightarrow$ Γ*X*Δ

This universal grammar is structurally complete. The reason for this is that any bracketing of category symbols is equivalent to every other bracketing, due to the rules for Product Associativity. Therefore, the grammar presented is the parenthesis-free AB-grammar.

In the next section we will discuss the language specific rules for Dutch.

3.2.4 Language specific rules

The grammar of each language is supposed to consist of universal rules and language specific rules. The language specific rules are distinguished from the universal rules by the possible use of constants in the grammar rules, and by other restrictions on the rules. The universal rules only allow for variables in the rule schemata. The language specific rules, however, can sometimes demand

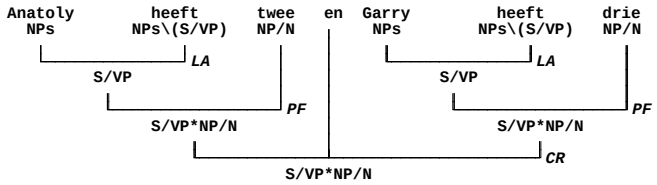
constants in the rule schemata. Besides restrictions on what category some variable X must be, the language specific rules of the grammar may also have restrictions that specify what categories cannot be instantiated in X.

The first language specific rule to discuss is the *Coordination Rule*. This rule accounts for the fact that two expressions of category X can be put together by a conjunction. The variable X can be any category symbol, including product sequences (X_1*...*X_n) of categories.

13. **Coordination Rule (CR)**
X^+ **en** X $\Rightarrow$ X

The Kleene plus on the left conjunct indicates that the conjunction can be preceded by one or more expressions of category X. The Coordination Rule accounts for the underlined part of the analysis of example (11), *Anatoly heeft twee en Garry heeft drie*. I present the analysis of this part in example (14).

14.



Anatoly has two and Garry has three

The Coordination Rule doesn't allow for embedded reduction, so there is no left-to-right or right-to-left derivation for sentences with coordinations. That means, structural completeness of the grammar is lost, when the Coordination Rule makes its entrance.

As we discussed in the previous chapters, Dutch shows disharmonic structures in the clustering of verbs. In subordinate clauses, the auxiliary verbs are right looking functors (over main verbs), whereas for example transitive main verbs are left looking functors (over objects). The grammar of Dutch should be able to account for these clustering phenomena. Within a categorial framework, we discussed the possibility of providing a lexical account of verb clustering, in chapter 1. But we have also seen that composing these clusters with VP modifiers

calls for a syntactic rule of Disharmonic Composition. In chapter 2, we mentioned the efforts of Bouma and Van Noord (1994) and Van Noord and Bouma (1994) to treat modifiers as arguments of the verbs. The impact on the PACG would be that the rule for Disharmonic Composition would possibly become superfluous. The general approach, however, is to treat modifiers as functions (X/X or X\X). In my analyses I will follow this approach.

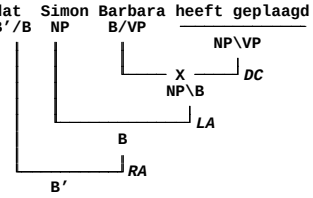
The *Disharmonic Rules* are presented in (15). The first disharmonic rule states the Disharmonic Composition rule in its basic form. The second is the Product Disharmonic Composition. In the definitions, X cannot be equal to B, otherwise the grammar would recognize subordinate clauses with the direct object preceding the subject, like in (16a). As predicted by the lexical account of verb clustering, VP modifiers cannot break up verb clusters. But as we see in (16b), (16c), (17) and (18), several other positions of such modifiers yield grammatical results.

15. **Disharmonic Rules**

Disharmonic Composition (DC)
$X/VP \quad Y_i\backslash(...\backslash(Y_n\backslash VP)) \quad \Rightarrow \; Y_i\backslash(...\backslash(Y_n\backslash X))$, where $X \neq B$

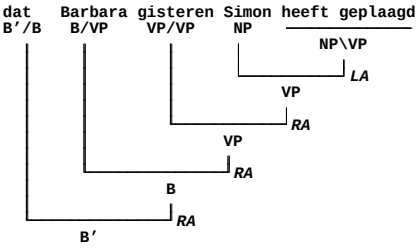
Product Disharmonic Composition (PD)
$\Gamma^*(X/VP^*Y_i\backslash(...\backslash(Y_n\backslash VP))^*\Delta \quad \Rightarrow \; \Gamma^*Y_i\backslash(...\backslash(Y_n\backslash X))^*\Delta$, where $X \neq B$

16.a * ...dat Simon Barbara heeft geplaagd



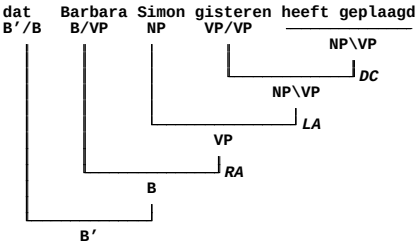
'that Simon Barbara has teased'
that Barbara has teased Simon

16.b

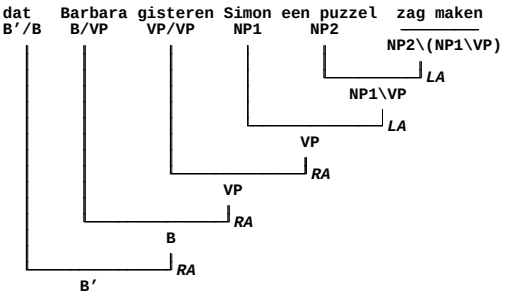


‘that Barbara yesterday Simon has teased’
that Barbara teased Simon yesterday

16.c

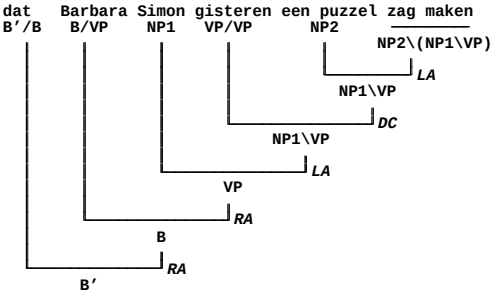


17.a

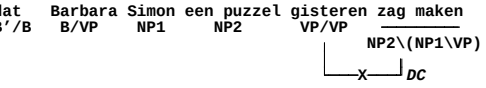


‘that Barbara yesterday Simon a puzzle saw make’
that Barbara saw Simon make a puzzle yesterday

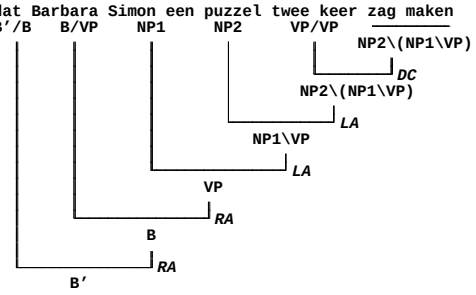
17.b



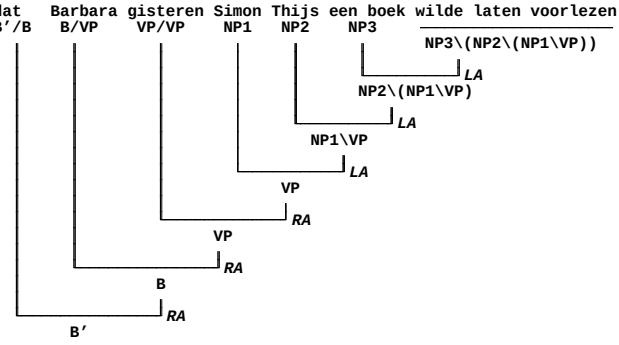
17.c * ...dat Barbara Simon een puzzel gisteren zag maken



17.d ...dat Barbara Simon een puzzel twee keer zag maken

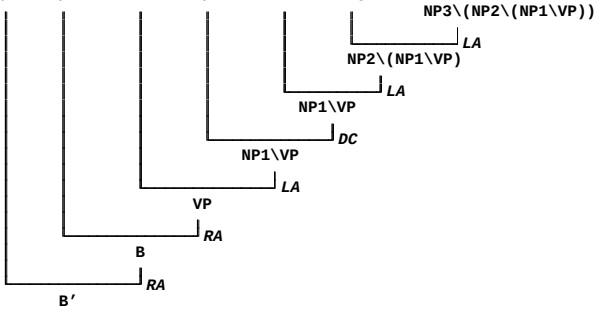


18.a ...dat Barbara gisteren Simon Thijs een boek wilde laten voorlezen

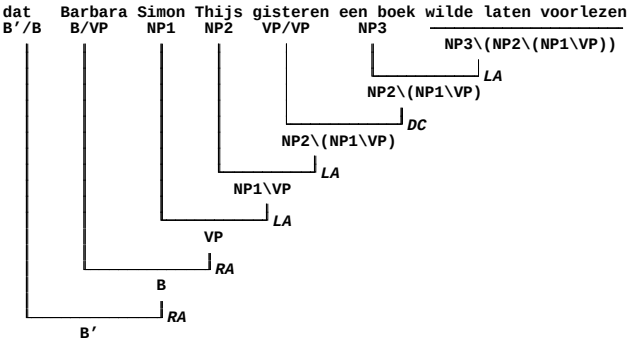


‘that Barbara yesterday Simon Thijs a book wanted let read out’
that Barbara wanted to let Simon read out a book to Thijs yesterday

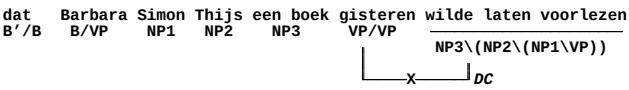
18.b ...dat Barbara Simon gisteren Thijs een boek wilde laten voorlezen



18.c



18.d *



The above observed phenomena appear to be puzzling, and hard to account for. Sentence (17c) seems to justify a restriction on the complexity of the subordinate functor. So the hypothesis would be that in definition (15) **n** is at most one. But this hypothesis must be rejected on the ground of the grammaticality of (17d) and (18c), for which Disharmonic Composition is used with n=2. What the ungrammatical sentences (17c) and (18d) have in common is, that the verb cluster first must apply to the direct object before Disharmonic Composition can be applied. But unfortunately, this fact cannot be captured in a rule scheme, because of the grammaticality of example (16c).

Besides the account for the word order in the examples (16), (17) and (18) the grammar straightforwardly excludes analyses of the ungrammatical word orders in (19), (20) and (21). In these examples, noun phrase arguments of the verbs disturb the verb clustering. In Houtman (1984), I proposed to use the feature [±LEX] to indicate that the verb phrase arguments shouldn't first combine with their noun phrase arguments, but instead should be applied to the VP-functors first. *Laten* would be marked (NP1\VP)/VP[+LEX], which means the argument has to be a lexical item. Such a feature would be superfluous in a lexical account of verb clustering, that is, before the syntactic component is invoked. In that case, we don't have to appeal to this rather ad hoc feature.

19. * dat Barbara heeft Simon geplaagd
'that Barbara has simon teased'
that Barbara has teased Simon

20. * dat Barbara zag Simon een puzzel maken
'that Barbara saw Simon a puzzle make'
that Barbara saw Simon make a puzzle

21. * dat Barbara wilde Simon laten Thijs een boek voorlezen
'that Barbara wanted Simon let Thijs a book read out loud'
that Barbara wanted to let Simon read out loud a book to Thijs

It is interesting to note that by adding Disharmonic Composition to a categorial grammar, the power of that grammar doesn't exceed context-freeness. As the Dutch mathematician Peter Zinn[38] proved, the AB-grammar supplemented with Disharmonic Composition is equivalent to Context Free Grammar. The universal part of the PACG is the parenthesis-free, or associative, AB-grammar, which is more powerful than the unassociative AB-grammar. According to Cremers (p.c.) supplying the associative AB-grammar with Disharmonic Composition would lead to a mildly context sensitive grammar.

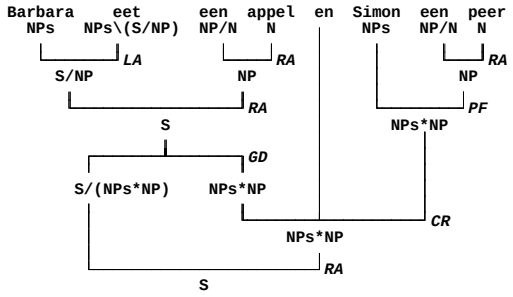
The third language specific rule for Dutch is the *Gapping Decomposition*[39] rule (GD) (see (23)). This rule makes an account of Gapping as non-constituent coordination possible. The first conjunct, that is analyzed as an S, can on the basis of GD be decomposed into a string X (= X_1*...*X_n) preceded by a rightward oriented function S/X. The string X can be chosen in such a way that it matches the categories of the expressions in the second conjunct. Accordingly, we can coordinate the two strings and yield a complex string of category X (= X_1*...*X_n), which then functions as the argument of S/X. We see the GD-rule exemplified in (22).

---

[38] Cf. Zinn (1993: 89-90).

[39] This rule was first introduced by Mark Steedman in a somewhat different fashion, as The Left Conjunct Revealing Rule.
The Left Conjunct Revealing Rule (< decompose)
$S \Rightarrow Y\ Y\backslash S$
Steedman (1990: 250)

22.



Barbara eats an apple and Simon a pear

23. **Gapping Decomposition (GD)**
S $\Rightarrow$ S/X X

I present the language specific rules for Dutch in (24). We will, in greater detail, go into Gapping phenomena in section 3.5. In that section we will compare the PACG-analysis of Gapping with the analysis of Morrill and Solias (1993). Furthermore we will look into combined Gapping/Right Node Raising cases.

24. **Dutch Rules**

Coordination Rule (CR)
X^{+} **en** X $\Rightarrow$ X

Disharmonic Rules

Disharmonic Composition (DC)
X/VP $Y_i\backslash(...\backslash(Y_n\backslash VP))$ $\Rightarrow$ $Y_i\backslash(...\backslash(Y_n\backslash X))$, where $X \neq B$

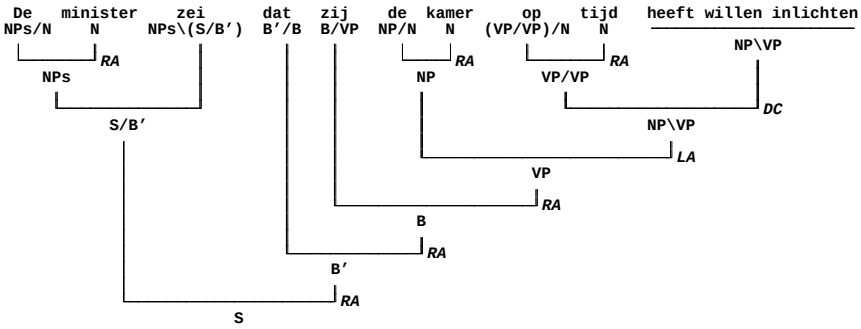
Product Disharmonic Composition (PD)
$\Gamma^*(X/VP^*Y_i\backslash(...\backslash(Y_n\backslash VP)))^*\Delta$ $\Rightarrow$ $\Gamma^*Y_i\backslash(...\backslash(Y_n\backslash X))^*\Delta$, where $X \neq B$

Gapping Decomposition (GD)
S $\Rightarrow$ S/X X

The above rule schemata behave differently from the categorial calculi, like the Lambek-calculus. Categorial calculi consist of axioms and inference rules. On the basis of these, it can be proved, whether a sequence of categories can be reduced to S. In other words, with the axioms and the inference rules, other theorems can be deduced.

The grammar defined here, on the other hand, must be regarded as a set of rules, from which no other rules are to be derived. This means that a sequence of categories can only be reduced to S by a finite number of applications of the rules of the grammar. However, we cannot demand that every rule application brings us closer to the final result, because the grammar includes at least one rule that can cause a derivational loop, and that is the rule for *Product Associativity*, presented in (8) and (12d). But, by definition, for each grammatical sentence there exists a finite derivation. As an example, we repeat sentence (34) of chapter 2 in (25).

25.



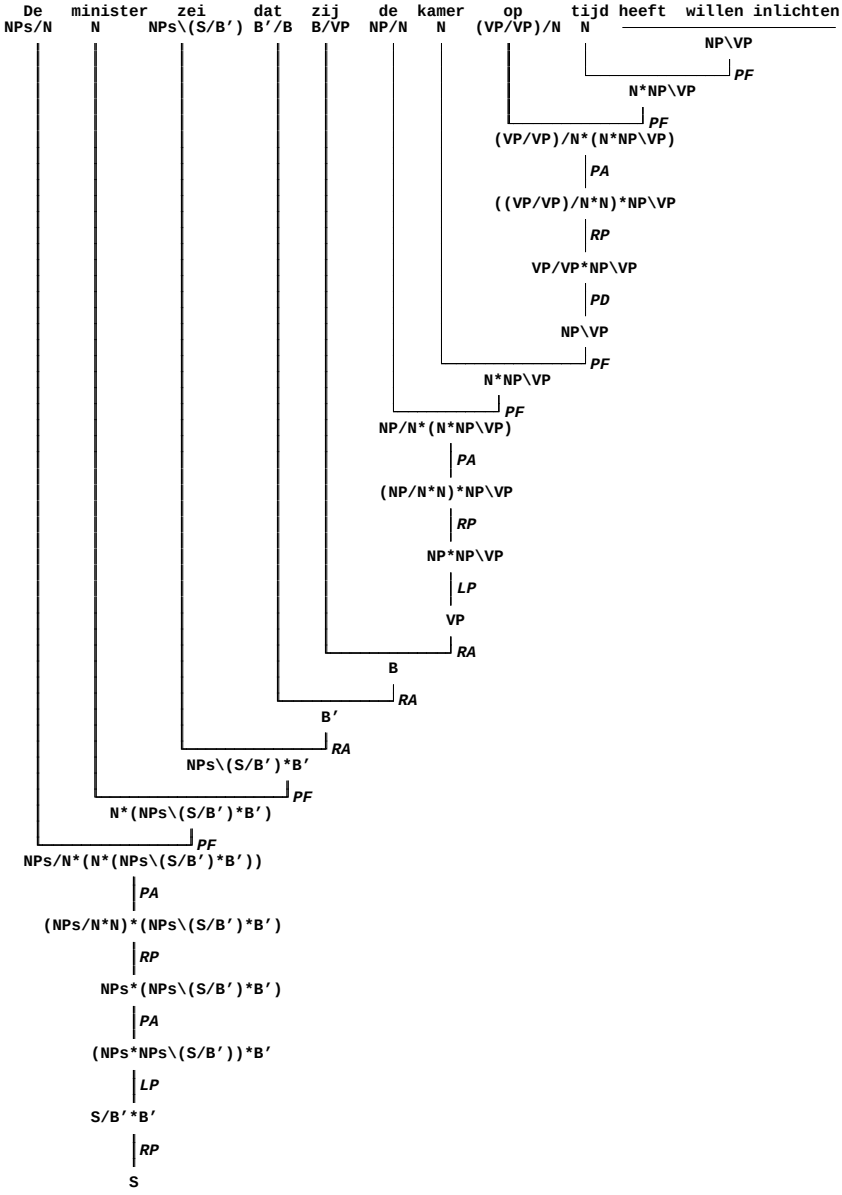
'The minister said that she Parliament on time has wanted to inform'
The minister said that she intended to inform Parliament on time

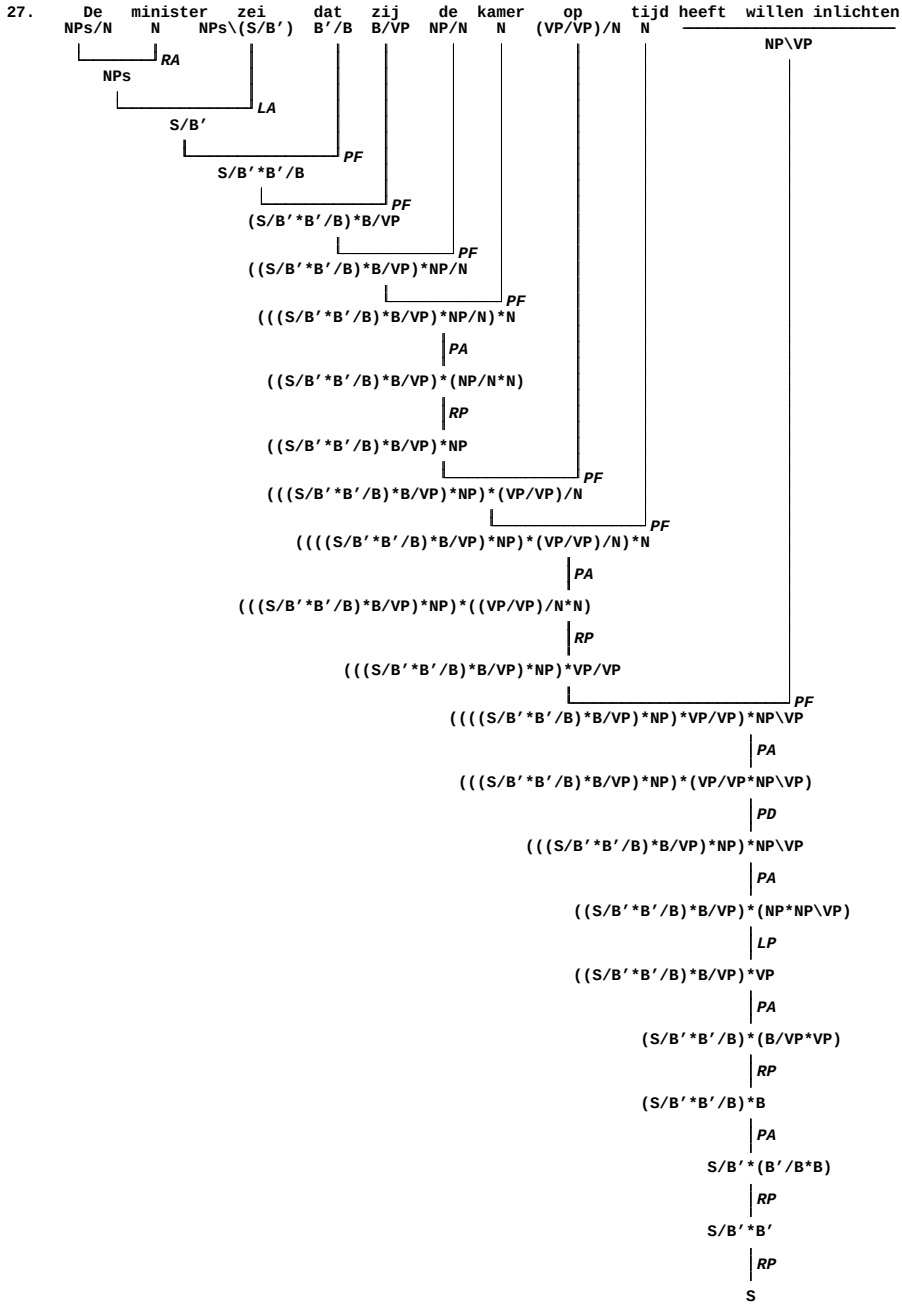
Because of the absence of Type Lifting rules and the Harmonic Composition rules, the PACG is less flexible than Lambek grammars. In the sense of ambiguities caused by these mechanisms, this is certainly true. On the other hand, part the universal rules of the PACG define a *core grammar* that is structurally complete. This means that for grammatical sentences each bracketing leads to a valid derivation. Completeness is lost in coordinated structures, because of the fact that the PACG doesn't contain a product version of CR. Structural completeness is also lost in the language specific Gapping Decomposition Rule. As an illustration of the structural completeness caused by the universal rules, I present the right-to-left analysis of (25) in (26), and the left-to-right analysis in (27).

As I mentioned earlier, structural completeness in the PACG is harmless, because -with or without the product rules- the only possible way to reduce strings of categories is by reduction through application (and Disharmonic Composition).

Also with respect to the notion *constituency* the structural completeness of the *core grammar* is harmless, as we will see in section 3.4.

26.





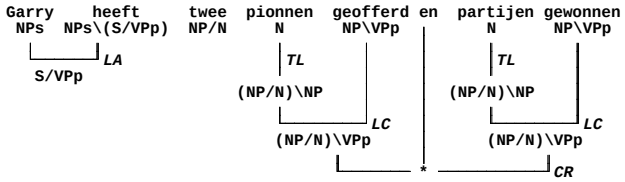
After this presentation of the PACG, it is worth examining how the coordination phenomena which lead to the refutation of the Lambek-calculus as descriptively

adequate for Dutch coordination, are accounted for in this grammar. In section 3.2.5 I discuss the combined Forward Conjunction Reduction / Right Node Raising cases, which caused the restrictions on the Coordination Rule of section 2.3.4. Within the Lambek-calculus the restrictions didn't lead to underivability of the ungrammatical sentences. We will see that with some minor alterations, the conditions can effectively block the ungrammatical sentences.

3.2.5 Conditions on coordination

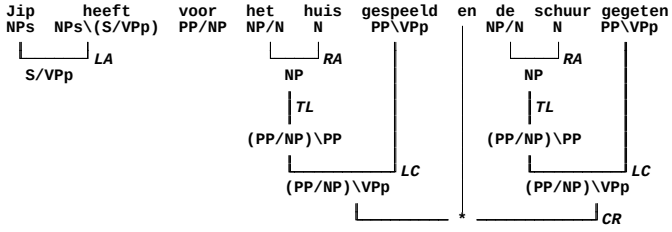
As I showed in Houtman (1987a) and Houtman (1987b), and as we have seen in section 2.3.4, restrictions have to be formulated on Forward Conjunction Reduction. In Houtman (1987a) I claimed that witness the examples (28) and (29), the Coordination Rule had to be restricted as in (30). The framework this claim was made in is the flexible F-grammar. As we have seen in chapter 1, the rules for Functional Composition and Type Lifting are part of this grammar.

28. *



'Garry has two pawns sacrificed and games won'
Garry has sacrificed two pawns and won two games

29. *

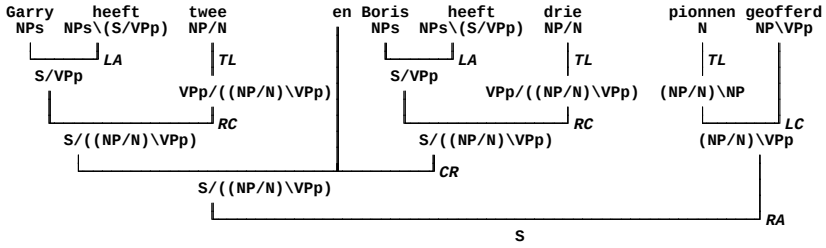


'Jip has before the house played and the barn eaten'
Jip has played in front of the house and eaten in front of the barn

30. **Coordination Rule 1987**
X **en** X $\Rightarrow$ X, where X $\neq$ (NP/N)\Y and X $\neq$ (PP/NP)\Y

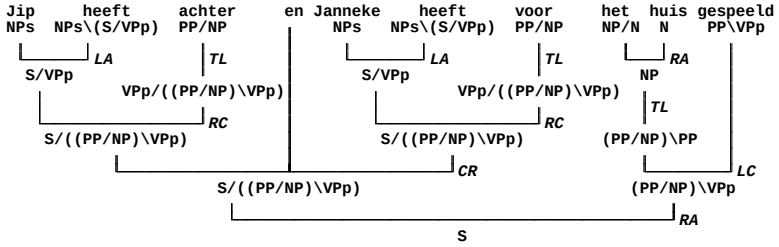
The reason the restriction couldn't be laid upon the rule for Type Lifting is the grammaticality of the Right Node Raising examples (31) and (32).

31.



'Garry has two and Boris has three pawns sacrificed'
Garry has sacrificed two pawns and Boris has sacrificed three pawns

32.



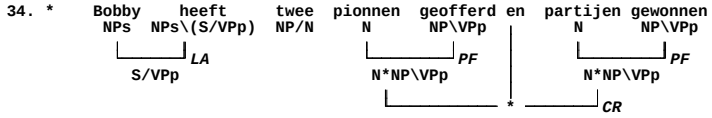
'Jip has behind and Janneke has for the house played'
Jip has played behind the house and Janneke has played in front of the house

We argued in chapter 2, that restrictions like the ones in (30) are not effective in flexible categorial grammars, like the Lambek-calculus, or for that matter, the F-grammar. This is one of the reasons we abandoned the Lambek-calculus as a descriptive device for Dutch. Accordingly, we formulated the PACG in the sections 3.2.3 and 3.2.4. Because of the absence of rules for Functional Composition and Type Lifting in the PACG, we have to reformulate the conditions on coordination (see (33)). The examples (28) and (29) will be presented in the PACG analysis in (34) and (35a).

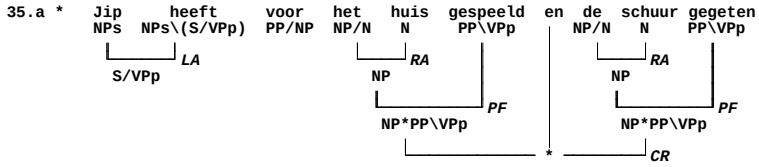
33. **Coordination Rule**

X^{+} **en** X $\Rightarrow$ X, where X ≠ N*NP\Y and X ≠ NP*PP\Z

34. *

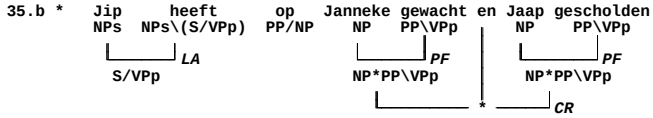


35.a *



There is no overall consensus on the analysis of (35a). Some consider the adjuncts *voor het huis* and *voor de schuur* PP-arguments of the verb, but others rather prefer the modifier analysis of these sequences, that is, the VP/VP-reading. In that case the argument for the condition on the Coordination Rule would no longer be valid. But considering non-adjuncts we can maintain the condition that X≠NP*PP\VP (example (35b)).

35.b *



'Jip has on Janneke waited and Jaap called names'
Jip has been waiting for Janneke and abusing Jaap

In the next section we will pay attention to phenomena referred to with the notion *Across-the board dependencies*.

3.2.6 Across-the-board dependencies

In categorial grammar functions apply to arguments. One of the most basic assumptions is that arguments are only bound once. So the function X/Y (or Y\X) only binds one instance of a Y. In Williams (1975) and Ross (1967), attention was paid to phenomena of double binding in the so called Across-the-board

dependencies (ATB). In these cases, one relative pronoun or wh-element binds two traces. Gazdar successfully treated these phenomena in a transformationless grammar (Gazdar (1981)). I will show in this section, that ATB-dependencies can be handled straightforwardly within the PACG-framework. ATB-dependencies are illustrated in the examples (36) and (37).

36. a. Ik zag een filmster die alle mensen kennen en weinig mensen aardig vinden
I saw a movie star who all people know and few people like
b. Ik zag een filmster die veel mensen kent en weinig mensen aardig vindt
I saw a movie star who knows many people and likes few people
* c. Ik zag een filmster die alle mensen kennen en weinig mensen aardig vindt
I saw a movie star who all people know and likes few people

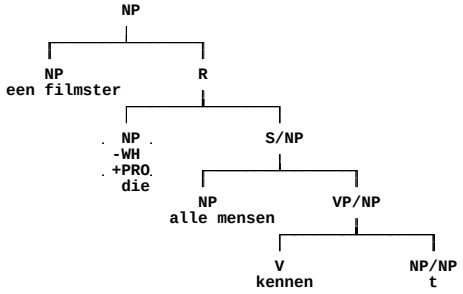
37. a. Ik vraag me af welk slachtoffer de overvallers hebben gegijzeld en de agenten hebben bevrijd
I wonder which victim the assailants took hostage and the policemen rescued
b. Ik vraag me af welk slachtoffer de verdachten heeft gezien en de inbraak heeft gehoord
I wonder which victim saw the suspects and heard the burglary
* c. Ik vraag me af welk slachtoffer de agenten hebben bevrijd en de inbraak heeft gehoord
I wonder which victim the policemen rescued and heard the burglary

In a transformational account, the relative pronoun in (36) binds two traces; one in the scope of *alle mensen kennen (veel mensen kent)* and one in the scope of *weinig mensen aardig vinden (vindt)*. In his article 'Unbounded dependencies and coordinate structure' (Gazdar (1981)), Gazdar treats such a trace as an expression of derived category X/X, that is as an X which lacks an X.[40] Furthermore, he

[40] Gazdar points out:
"This notion is reminiscent of categorial grammar but, despite a tenuous conceptual link, these derived categories are not to be interpreted in the way categorial grammar prescribes. The intended interpretation is as follows: a node labeled α/β will dominate subtrees identical to those that can be dominated by α, except that somewhere in every canonical subtree of the α/β type there will occur a node of the form β/β dominating a resumptive pronoun, a phonologically null dummy element, or the empty string, and every node linking α/β and β/β will be of the σ/β form. Intuitively, then, α/β labels a node of type α which dominates material containing a hole of type β (i.e. an extraction site on a movement analysis). So, for example, S/NP is a sentence which has an NP missing somewhe-

analyzes the expressions with a gap by ordinary coordination. Thus, the gap in (36a) and (37a) is of the NP type for *alle mensen kennen*, *weinig mensen aardig vinden*, *de overvallers hebben gegijzeld*, and *de agenten hebben bevrijd*. In (36b) and (37b), the gap is also of the NP-type. By a metarule Gazdar transforms the remaining S/NP in VP. By doing this, he can discriminate between expressions with a subject gap and expressions with an object gap. And for the examples (36c) and (37c) it is now easy to see that expressions of type S/NP cannot be coordinated with expressions of type VP. As an illustration, I present a part of Gazdar's analysis of sentences like (36a) in (38).

38.



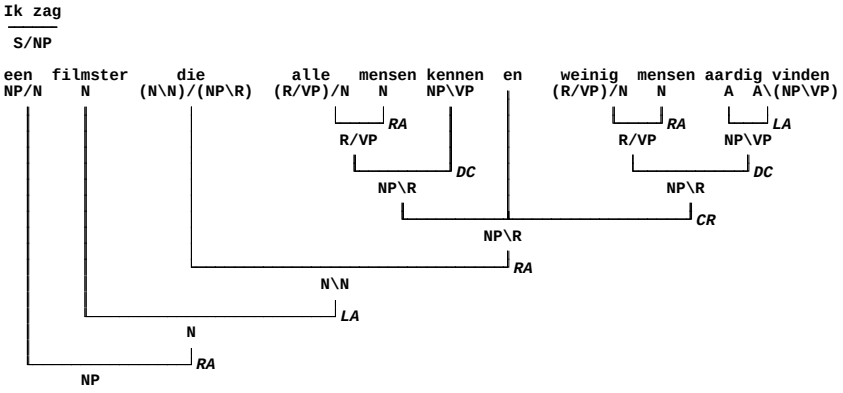
a moviestar who all people know

Within the PACG presented in this chapter, a straightforward analysis of the ATB dependencies is possible, as far as peripheral extractions like in (36) and (37) are concerned. Reminiscent of Gazdar's metarule, we only need to assign two different categories to the relative pronoun in sentence (36); the category (N\N)/(NP\R) for relative pronouns that function as objects in the embedded clause, and the category (N\N)/VP for relative pronouns that function as subjects in the embedded clause. For wh-elements in embedded questions, we also need two categories; (QB/(NP\QB))/N for objects, where QB stands for embedded question, and (QB/VP)/N for subjects. From the PACG-analyses it follows that coordination of the two expressions is prohibited in (36c) and (37c), presented below as (39c) and (40c), simply because the two conjuncts don't have the same category.

---

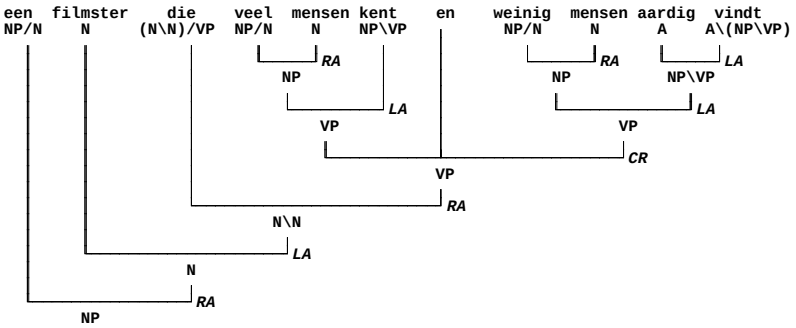
re." Gazdar (1981: 159)

39.a **Ik zag**



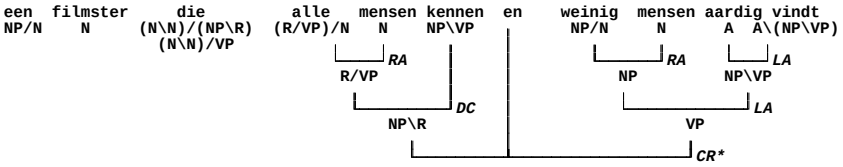
I saw a moviestar who all people know and few people like

39.b **Ik zag**



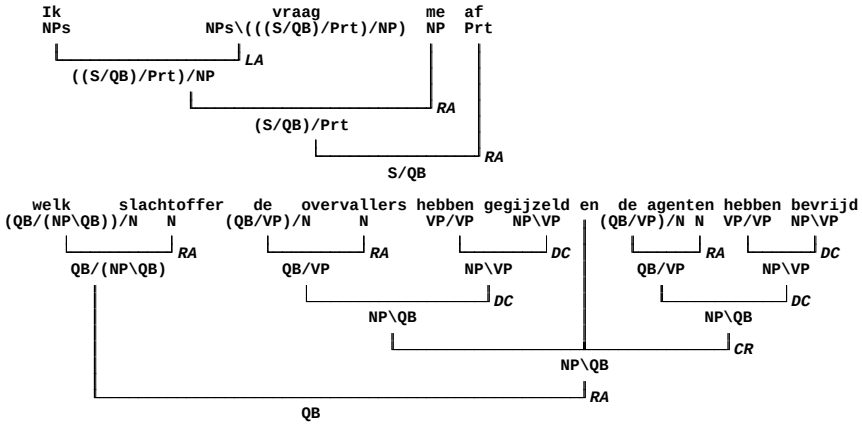
I saw a moviestar who knows many people and likes few people

39.c * **Ik zag**



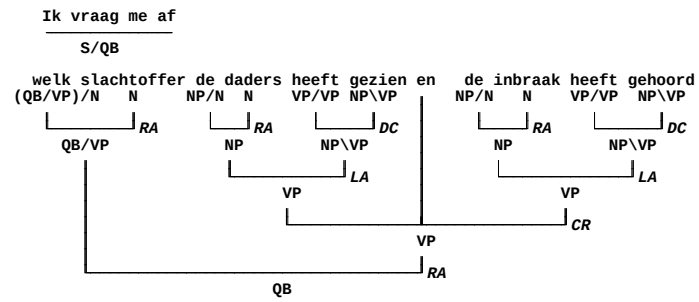
I saw a moviestar who all people know and likes few people

40.a



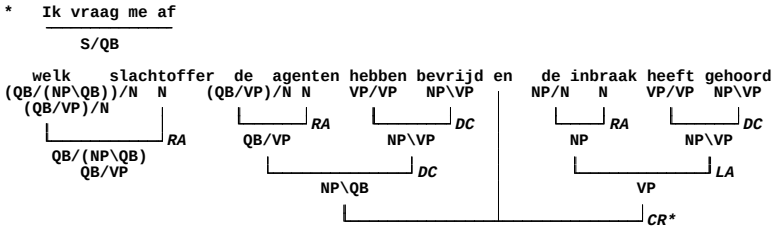
I wonder which victim the assaulters have taken hostage and the policemen have released

40.b



I wonder which victim has seen the assaulters and has heard the burglary

40.c *



I wonder which victim the policemen have released and has heard the burglary

We now turn to some concluding remarks with respect to section 3.2.

3.2.7 Concluding remarks

In section 3.2 we presented the product-based applicative categorial grammar (PACG) for Dutch. The universal part of this grammar consists of application rules and product rules. These rules are all provable theorems of the Lambek-calculus. Therefore, this universal grammar can be considered a subset of the Lambek-calculus. But as we remarked in section 3.2.3, the universal grammar is the associative or parentheses-free AB-grammar.

More telling than the resemblances, are the differences between the PACG and the Lambek-calculus.

The Lambek-calculus is a proof system. On the basis of axioms and inference rules, other rules are provable theorems of the system. In chapter 1, we discussed several of these theorems. Among them Functional Composition, Type Lifting, the Geach Rule, several other Division rules, Strong Composition and Dekker's paradox. The PACG, on the other hand, is a grammar where all rules are specified and no other rules can be inferred. The PACG, thus, is a grammar that builds up derivation trees.

The PACG is less flexible than the Lambek-calculus with respect to the use of reduction rules and structure building rules. Examples of the last kind are the Type Lifting rules that are part of the Lambek-calculus. These rules are totally excluded from the PACG. The same holds for the rules for Harmonic Functional Composition. This means that reduction to S only takes place by means of the application rules and the derived application rules. In other words, there is only one way to reduce sequences, and that is to apply functions to arguments. One exception must be made for the disharmonic structured verb clusters in Dutch. For this phenomenon we need a syntactic rule for Disharmonic Composition, even if we should consider verb clustering a lexical phenomenon, as we discussed in chapter 1.

Part of the flexibility that is lost by not allowing rules like Functional Composition and Type Lifting is regained by the product rules. Especially, the Product Associativity rule makes it possible to compute a category for any two adjacent expressions. But different from the Lambek-calculus, the PACG doesn't necessar-

ily compute a product-free category for such sequences. The flexibility in the PACG is harmless with respect to function/argument structures.

The PACG makes it possible to reexamine the meaning of the notion *constituent*, on the basis of product categories and product-free categories. We will do so in section 3.4.

Finally, in 3.2.5 we argued that conditions on categorial rules are possible within the PACG, while this was evidently impossible within the Lambek-calculus.

In the next section, I will compare the PACG to Mary Wood's *Thursday Grammar*, a grammar that also makes a substantial use of product categories. Before doing this, I will present the PACG in its current form.

41. **Product-based Applicative Categorial Grammar (PACG)**

Universal Rules

Reduction Rules

Right Application (RA)
X/Y Y ⇒ X
Left Application (LA)
Y Y\X ⇒ X

Product Rules

Product Formation (PF)
X Y ⇒ X*Y
Product Associativity (PA)
(X*Y)*Z ⇔ X*(Y*Z)

Derived Reduction Rules

Right Product Application (RP)
Γ*((X/Y)*Y)*Δ ⇒ Γ*X*Δ
Left Product Application (LP)
Γ*(Y*(Y\X))*Δ ⇒ Γ*X*Δ

41. **Dutch Rules**

Coordination Rule (CR)

X^{+} **en** X $\Rightarrow$ X, where $X \neq N^{*}NP\backslash Y$ and $X \neq NP^{*}PP\backslash Z$

Disharmonic Rules

Disharmonic Composition (DC)

$X/VP \quad Y_i\backslash(...\backslash(Y_n\backslash VP)) \Rightarrow Y_i\backslash(...\backslash(Y_n\backslash X))$, where $X \neq B$

Product Disharmonic Composition (PD)

$\Gamma^{*}(X/VP^{*}Y_i\backslash(...\backslash(Y_n\backslash VP)))^{*}\Delta \Rightarrow \Gamma^{*}Y_i\backslash(...\backslash(Y_n\backslash X))^{*}\Delta$, where $X \neq B$

Gapping Decomposition (GD)

S $\Rightarrow$ S/X X

3.3 Wood's Thursday Grammar

In this section, I will discuss some of the aspects of Wood's categorial grammar for coordination.[41] She christened her grammar *Thursday Grammar*. The most important feature of this grammar is the use of the product operator for the account of coordination phenomena.[42] Although the product operator is implicitly present in categorial grammars in general, Wood is the first to apply the product operator with a specific goal, namely to restrict the grammar to a minimal set of descriptive tools, not tolerating flexible rules, as Type Lifting and Generalized Composition. Generalized Composition differs from Harmonic Functional Composition, discussed in chapter 1, in that the denominator of the first function doesn't have to equal the numerator of the second function, but that the numerator is reducible to that category. Y$ are the function categories

---

[41] Wood (1988).

[42] Another, more formal, approach to product-based systems is discussed in Oehrle (1987).

reducible to Y. Generalized Composition is formulated in (42). An example is supplied in (43). This example is an instance of Strong Composition within the Lambek-calculus, discussed in chapter 1.

42. **Generalized Composition**
X/Y Y$/Z ⇒ X$/Z

43. X/Y (Y/U)/Z ⇒ (X/U)/Z

The product operator avoids overgenerating operations like Type Lifting. In combination with Generalized Composition, Type Lifting would cause extreme overgenerating power.

Wood exploits the following product rules.

44. **Product Formation**
X Y ⇒ X*Y

45. **Deconstruction**
X*Y ⇒ X Y

46. **Infixing**
A B**I**(A*C) C ⇒ B

47. **Flattening**
(X/Y)/Z ⇒ X/(Z*Y)

Rule (44) is obvious. Two expressions of category X and Y can form a complex expression of category X*Y. As Wood notes:

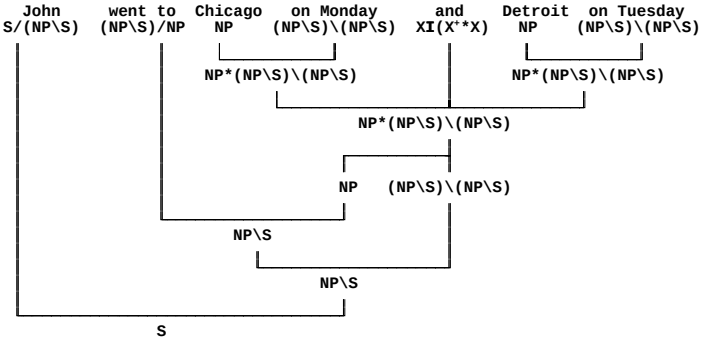
> "A product in a derivational categorial calculus is an ordered pair of categories which functions as a single (complex) category."
> (Wood (1988: 76))

Rule (45) is less obvious. Wood needs this rule to account for sentences like (48) below. Although Wood generally puts the subject (S/(NP\S)) and the finite transitive verb ((NP\S)/NP) together by Forward Composition, resulting in an

expression of category S/NP, this strategy doesn't work in example (48). Here, the category for the transitive verb *went to* must first apply to the argument category, in order to provide the right argument for the modifier of category (NP\S)\(NP\S). But therefore, the product category NP*(NP\S)\(NP\S) has to be deconstructed.

Rule (46) is also exemplified in sentence (48),[43] where the conjunction is a special instance of B**I**(A*C), namely X**I**(X^{+}*X). The operator **I** infixes the conjunction in a sequence of expressions of category X. There may be an arbitrary number of expressions of category X to the left of the conjunction -with a minimum of one- as indicated by the Kleene star. But there can only be one -and it must be exactly one- expression of category X to the right of the conjunction, thus accounting for sequences like *John, Mary, Peter and William*.

48.



The flattening rule (47) can, for example, change the category of a ditransitive verb to a function category with only one argument. The advantage of this rule is that the subject and the tensed verb can be composed by simple forward composition, instead of by some sort of generalized forward composition. Wood puts it as follows.

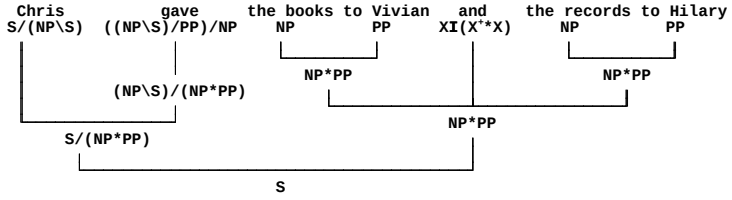
> "Where both are included [the product rule and the flattening rule, JH] in a grammar (...) the need for fully generalized composition is greatly reduced or possibly eliminated altogether."
> Wood (1988: 77)

---

[43] Wood (1988: 225), example 4.3.61e.

Furthermore, it takes the two arguments as one complex argument. Example (49) below is taken from Wood (1988: 218), example 4.3.53.

49.



According to Wood the conjunction is the head functor in coordinations. The category X**I**(X^{+}*X) provides a flat structure for coordinations, which is desirable according to Wood. Hereby, she rejects "a nested, hierarchical category scheme (X|X)|X for conjunctions; which is exactly appropriate for subordinating, but not for coordinating conjunction". (Wood (1988: 168-169))

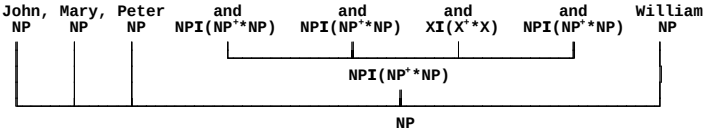
She also rejects the coordination rule X **conj** X ⇒ X, because she claims PS-rules are not consistent with categorial grammars and therefore shouldn't be incorporated in them.

I would like to make a few remarks about her proposal. In the first place the approach with a product formation rule, incorporated in the categorial grammar, seems to be very promising. It can rule out (Harmonic) Functional Composition and Type Lifting entirely, as we discussed in section 3.2. Advantages of such an approach are the fact that it avoids overgeneration, and the possibility of formulating conditions on categorial rules. Furthermore, as we will discuss in section 3.4, the product approach gives us the opportunity to reinforce the notion *constituent* in categorial grammar, which would not be possible in categorial grammars with Functional Composition. Wood doesn't exclude (Harmonic) Functional Composition. Therefore, her *Thursday Grammar* is less compatible with the claims I make in section 3.4 on *constituency*.

There are other objections to specific parts of her approach. The first objection concerns the category for conjunctions, or to use a category for conjunctions at all. When using the category X**I**(X^{+}*X) for conjunctions, we cannot avoid conjunctions themselves being coordinated. The disadvantage of using a category for conjunctions is that derivations can be made with intermediate results. The

syncategorematic account, proposed in this study, excludes this. I illustrate this in example (50), where X is short for NP**I**(NP^+*NP).

50. *



It is obvious that sentence (50) cannot be derived in the PACG, because the syncategorematic rule X^+ **and** X ⇒ X (comparable to the Dutch Coordination Rule X^+ **en** X ⇒ X) requires on both sides of the conjunction an expression of category X.

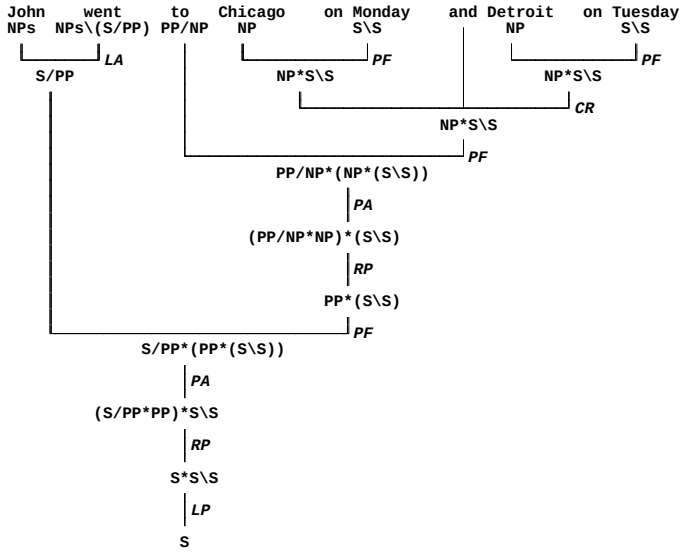
The category Wood uses for coordinations doesn't fit flat multiple coordinations, exemplified in (51). The coordination rule I use above doesn't cover these cases either. One could think of using the Kleene plus to account for flat multiple coordinations, as in rule (52).

51. Chris and Robin and Geese and ducks and swans

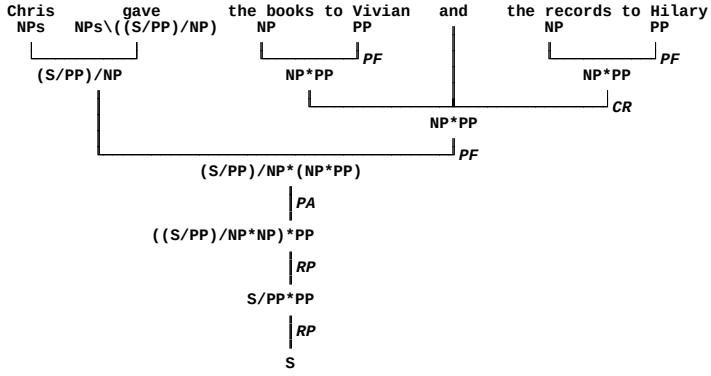
52. **Flat Multiple Coordination**
(X and)$^+$ X ⇒ X

The second objection is that, in my opinion, two rules introduced by Wood are superfluous. The first is the flattening rule, and the second the deconstruction rule. If we adopt the PACG, we not only have access to the Product Formation rule, but also to the Product Associativity rules. These account for the derivation of the sentences (48) and (49), without making use of Flattening and Deconstruction (see the examples (53) and (54)). Notice that we categorize the adverbials *on Monday* and *on Tuesday* as S\S, instead of as (NP\S)\(NP\S), as Wood does.

53.

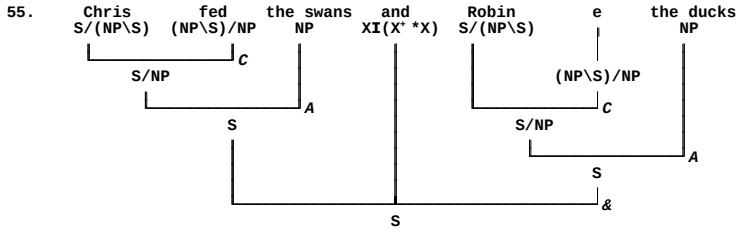


54.



The last remark about Wood's *Thursday Grammar* concerns her argumentation against Decomposition in Gapping cases. In her opinion Decomposition violates Steedman's Adjacency Principle. Instead, she proposes a rule of Reconstitution, which recovers the category of the missing element on the basis of the elements present in the first conjunct. The analysis of Gapping runs as follows.

55.



The disadvantage of this approach is the use of empty categories.[44] In general categorial grammars are supposed to supply natural language analyses without making use of empty categories. When one uses empty categories in the grammar, then at least two things must be specified: when and where they can turn up. It is not the case that for every element present in the first conjunct and absent in the second, an empty element can be introduced. We can see this in the following two examples. In (56), a perfectly grammatical Gapping sentence is presented, while the sequence in (57) is ungrammatical. Notice, that the PACG excludes sentence (57) on the grounds of the Gapping Generalization, stated in (81) in section 3.5.2.

56. Jan zei dat Klaas een boek aan Marie zou geven en **e e e e** een plaat **e e** zou terugvragen
'Jan said that Klaas a book to Marie would give and a record would ask back'
Jan said that Klaas would give a book to Marie and would ask back a record

---

[44] Oehrle (p.c.) notes two other fundamental problems to Wood's approach to Gapping. The first problem is semantic interpretation. The following sentence has two interpretations:

i. Jones can't live in Omsk and Smith in Tomsk

interpretation a: not(permissible(Jones live in Omsk AND Smith live in Tomsk))
interpretation b: (not permissible (Jones live in Omsk)) AND (not permissible (Smith live in Tomsk))

Analyses based on the deletion or reconstitution only get the second analysis in any natural way.

The second problem is recursiveness. It isn't too obvious how to formulate the reconstitution rule so that it works for cases like (ii):

ii. Jones can live in Omsk and Smith in Tomsk OR Smith in Omsk and Jones in Tomsk.

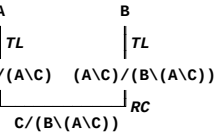
57. * Jan zei dat Klaas een boek aan Marie zou geven en Piet **e e** Kees een plaat **e e e e**
'Jan said that Klaas a book to Marie would give and Piet Kees a record'
Jan said that Klaas would give a book to Marie and Piet said that Kees would give a record to Marie

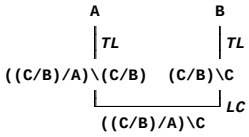
The reason for Wood to prefer this approach to Steedman's Decomposition approach was that Steedman had to compose the discontinuous parts of the second conjunct to a constituent. As we will see in section 3.4, the elements of the second conjunct are not composed as a constituent within the present PACG analysis, because categories containing the product operator are not considered constituents. This makes Wood's objections against Decomposition groundless.

3.4 Constituency revisited

3.4.1 Introduction

In this section, I will examine the notion *constituent* in categorial grammar. As we noticed in chapter 2 of this study, the notion *constituent* has become meaningless in flexible categorial grammars. Any two adjacent categories can form a constituent of some category, by means of Type Lifting rules and rules of Functional Composition. We have shown this in section 2.2.1, example (1). We will repeat this example in (58).

58. a.


b.


In this section, I will discuss two efforts to assign the notion *constituent* a substantive meaning. First, in section 3.4.2, we will discuss the notions *dependency constituent* and *maximal dependency constituent*, introduced by Barry and Pickering (1990).

To this approach some criticism has come up by Van der Linden (1993). In section 3.4.3 we discuss his approach to the notion *constituent* in a subsystem of the Lambek-calculus. Finally, in section 3.4.4, I present the notion *constituent* within the PACG. It appears that, on the basis of product categories, a natural distinction can be made between categories and constituents.

3.4.2 Barry and Pickering's dependency constituent

As far as flexible systems, like the Lambek-calculus, are concerned, the notion of *constituent* has become void. Attempts have been made to restore the meaning of the notion *constituent*. Barry and Pickering (1990) introduced the notion *dependency constituent*. Dependency constituents are strings that have dependency-preserving analyses. Barry and Pickering state:

> "We shall refer to a meaning representation as *dependency-preserving* if it maintains the head-dependent relations derivable from lexical assignments."
> Barry and Pickering (1990: 26)

Barry and Pickering identify head-dependent relations with function-argument structures. So, in other words, when the syntactic and semantic representation of a string is missing a functor, of which the arguments are in the string, then the string is not a dependency constituent. The consequences of this definition are, that the bracketed strings in (59) are dependency constituents, where the bracketed strings in (60) are not.

59. a. [the dog] runs
    b. [John likes] Mary
    c. John [will see] Mary

60. a. the [dog runs]
    b. I think [that Harry] left
    c. I showed [Mary John]

The string in (59a) is a complete NP, and the strings in (59b) and (59c) have missing arguments. The strings in the examples in (60) all have missing heads,

and are therefore not dependency constituents. Barry and Pickering use the notion of *dependency constituent* to characterize the class of conjoinable expressions.

The first thing they have to admit, however, is that coordination does not have to involve dependency constituents, and that dependency constituents do not always license coordination. In cases where the structure of the conjuncts is the same, coordination of dependency constituents is accounted for correctly. But more interesting are cases where the conjuncts have different structures. For example in (61), Barry and Pickering argue that the two bracketed strings have the same category, and yet are not conjoinable.

61. * John loves [Mary madly] and [Sue]

They categorize *loves* as ((NP\S)/AdvPost*)/NP, meaning that at first an NP-argument must be found to the right and then *zero or more* post adverbials can be applied (this is the meaning of the 'Kleene star' *, as opposed to the 'Kleene plus' $^{+}$, meaning *one or more*). After that, application to the subject NP yields an S. Both *Mary madly* and *Sue* receive the category (((NP\S)/AdvPost*)/NP)\(NP\S). In other words, they are strings searching for a verb like *loves* to yield an NP\S, a function that looks for a subject NP to form a sentence.

In my opinion there are two objections to this approach. If we adopt the view of Barry and Pickering that post adverbials are arguments instead of functions over verbs, then we have to admit that the instantiation of the category of *loves* should be different for the conjuncts *Mary madly* and *Sue*. As far as the first conjunct is concerned, *loves* acts as a ((NP\S)/AdvPost)/NP, while for the second conjunct *loves* acts as a (NP\S)/NP. In other words, we should discriminate between instantiations of the Kleene star and instantiations of the Kleene plus.

Secondly, although the 'Kleene star' suggests that we should generalize over these different cases, it seems that this is not correct, and the categories of *Mary madly* and *Sue* in fact differ. In the PACG I propose in this chapter, sentence (61) is ruled out by the fact that the coordinated strings do not have equal categories. The first conjunct could, if we use the categorisation of Barry and Pickering, be typed as NP*AdvPost and the second as NP.

The arguments for the ungrammaticality of the sentences (62) and (63) run accordingly. In (62), Barry and Pickering categorize both conjuncts as S/(NP\S), whereas in the underlying approach of this chapter the string *I believe that John* cannot be reduced to this category, and the category of *Mary* cannot be lifted to this category. The same applies to example (63), in which *two small* and *three* receive the category NP/(AdnPre*\N) in the approach of Barry and Pickering. That is, the function searches for a common noun, which can be preceded by zero or more adjectives. In the PACG presented here, adjectives are typed N/N, and common nouns N. By the absence of Functional Composition the string *two small* cannot be reduced to NP/N, but will by the PF-rule be typed NP/N*N/N which obviously differs from NP/N.

62. * [I believe that John] and [Mary] is a genius

63. * [two small] and [three] oranges

Barry and Pickering state that if two dependency constituents have the same category, they can be conjoined. This means that no restrictions are put upon the internal structure of the dependency constituents. But as Barry and Pickering note themselves:

> "But even when the conjuncts are not dependency constituents, there appears to be no restriction on the internal structure of any substring that is a dependency constituent."
> Barry and Pickering (1990: 34)

So the sentences in (65) and (66) are as good as the one in (64). On the basis of these examples, Barry and Pickering define the notion *Maximal Dependency Constituent* (MDC). A division of a string in maximal dependency constituents means that when the string Γ consists of the MDC's $\gamma_1,...,\gamma_n$, there is no such i that $\gamma_i\gamma_{i+1}$ is a dependency constituent. The division in MDC's enables us to identify the conjuncts as parallel structured strings.

64. John loves [Mary madly] and [Sue passionately]

65. John loves [Mary madly] and [the young woman passionately]

66. John loves [Mary madly] and [Sue with great ardour]

In section 3.4.4, I show that in the PACG such an analysis is possible, without introducing the notion *maximal dependency constituent*.

To the approach of Barry and Pickering some objections have been made by Van der Linden in his thesis.[45] We will address his objections in the following section, 3.4.3.

3.4.3 Van der Linden's dependency constituent

In his thesis Van der Linden (1993) argues for a notion of *dependency constituent* in type-theoretical terms. His main argument against Barry and Pickering is that their *dependency-preserving* property, which underlies the notion of dependency constituent, has no meaning in type logic. His claim is that you should create a subsystem of the Lambek-calculus **L**, in which only dependency constituents are recognized. The way to do this, is according to Van der Linden, by means of headed type constructors. For the same reason of having no type-theoretical status Van der Linden rejects the notion *maximal dependency constituent*.

Another objection Van der Linden has to the approach of Barry and Pickering, is the categorisation of pre- and post-modifiers. Barry and Pickering treat functions as heads, but he wants to treat modifiers [of functional category X/X or X\X, JH] as dependents. Therefore they are not typed as X/X and X\X respectively, but the functions are specified for zero or more of these modifiers. This leads to a complication in case we assume modifiers to the right as well as to the left. So, if an expression of category X has a premodifier and a postmodifier, we need two categorizations to account for scope-ambiguities: Xmod\(X/Xmod) and (Xmod\X)/Xmod. In case of two pre- and postmodifiers six categorizations are necessary, etc. Van der Linden argues that this treatment is in fact not necessary. He refers to Bach (1983), who defined the notion of *syntactical head* on the basis of function-argument structures. Bach stated that for endocentric functions A/A the argument is the head. For exocentric functions A/B, with A and B in the same

---

[45] Van der Linden (1993).

projection class, the argument is the head. For other exocentric functions the functor is the head. Applying this notion of head to the definition of dependency-preservation, Van der Linden claims:

> "A lambda-calculus (...) representation **a** is dependency-preserving iff it does not involve abstraction of a variable that occurs as a head within it."
> Van der Linden (1993: 80)

This alternative definition does not allow for introduction of a functor that is a head, but it allows for introduction of functors that are dependents. Furthermore, it does not allow for the introduction of arguments, if they function as heads.

Van der Linden follows Moortgat and Morrill (to appear) in defining headed variants of the reduction slashes and the product operator. He introduces the following operators:[46]

$/_h$	the headed variant of '/'	(the function is the head; the argument is the dependent);
$/_d$	the dependent variant of '/'	(the function is the dependent; the argument is the head);
$\backslash_h$	the headed variant of '\'	(the function is the head; the argument is the dependent);
$\backslash_d$	the dependent variant of '\'	(the function is the dependent; the argument is the head);
$*_l$	the head of the product is the left constituent;	
$*_r$	the head of the product is the right constituent.	

Van der Linden argues that, within the Lambek-calculus, these headed operators do not yield the required set of dependency constituents. On the one hand it is too strong, because it allows for introduction of headed functions. On the other hand, it is to weak, because in some configurations (X,(Y,Z)), (X,Y) cannot be considered a constituent, although this sometimes seems to be wished for. Van der Linden builds up a logic (HNL-), a variant of the so called *Headed Non-*

[46] Van der Linden (1993: 81) uses a notation for the headed operators in nice graphics. For the purposes of this book, the notation with subscripts suffices.

associative Lambek Calculus (HNL), in which the right rules for $/_d$ and $\backslash_d$ have no part, because they result in semantic representations with abstraction over heads. In some specific combinations of heads and dependents, he seems to need associativity. The restricted associativity he introduces leads to the system **D**. It is obvious that the way Van der Linden tries to restore the notion of *constituent* is quite different from the approach in this chapter. Furthermore Van der Linden does not illustrate the linguistic consequences of his logic, where coordination is concerned. Although the approach to constituency may be promising, I would have preferred linguistic examples to test the theory. I leave the comparison of the two approaches to further research, and turn now to the definition of *constituent* in the framework of the PACG presented here.

3.4.4 Constituency in the PACG

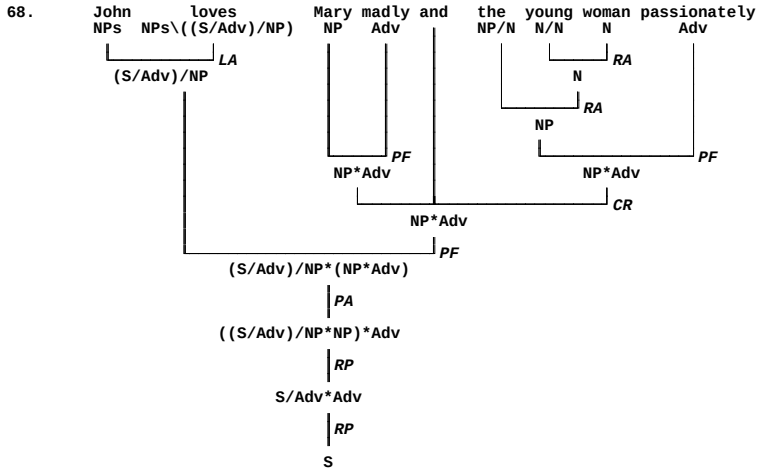
By using the PACG of section 3.2, we have the possibility of regaining the content of the notion *constituent* in a straightforward way. Like in the Lambek-calculus, constituent status is defined by the functional (i.e. applicative and compositional structure). Furthermore, product structures do not define constituent status.

67. **Constituency**
A word or a word sequence is a *constituent* if and only if its category is reducible to a product-free type

Definition (67) defines all lexical elements as constituents, as well as sentences. All intermediate results of reduction rules yield constituents, as far as no product operator is present in the category of this intermediate result.

This definition accounts in a natural way for the parallelism of sentence (65) above, without the need to define *dependency constituents* or *maximal dependency constituents*. This is illustrated in (68).

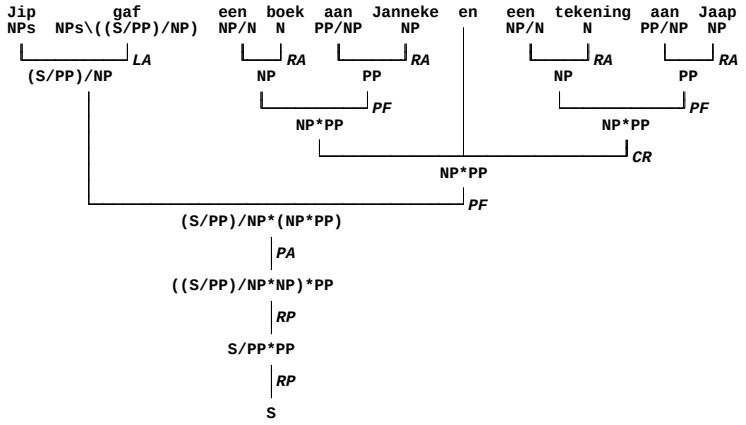
68.



We will now look at Forward Conjunction Reduction, Right Node Raising, and Gapping phenomena to find out in which cases constituents are involved, according to definition (67).

Forward Conjunction Reduction or *Left Node Raising* shows a conjunction of word sequences which needs to be completed to the left to make up a sentence. Example (69) shows that the sequences *een boek aan Janneke* en *een tekening aan Jaap* (NP*PP) are not constituents, and therefore the conjunction of these parts is not a constituent. It was these cases, among others, that Dowty (1988) referred to as *non-constituent coordination*. Within the categorial framework presented here, we can maintain this notion.

69.

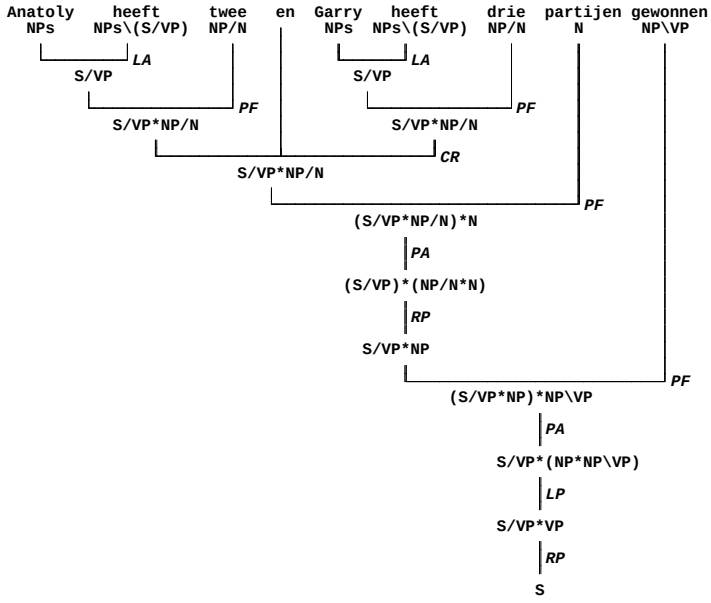


Jip gave a book to Janneke and a drawing to Jaap

In sentence (69), besides the lexical elements and the whole sentence, the following words and word sequences are constituents:
Jip gaf, een boek, een tekening, aan Janneke, aan Jaap.

When considering Right Node Raising we observe that there are no conditions whatsoever on the constituenthood of the ‘raised’ part or of the remainder of the sentence. Looking at example (70), we find according to the definition of constituency in (67) no constituents, besides the lexical elements, the combination subject / tensed verb, and the whole sentence.

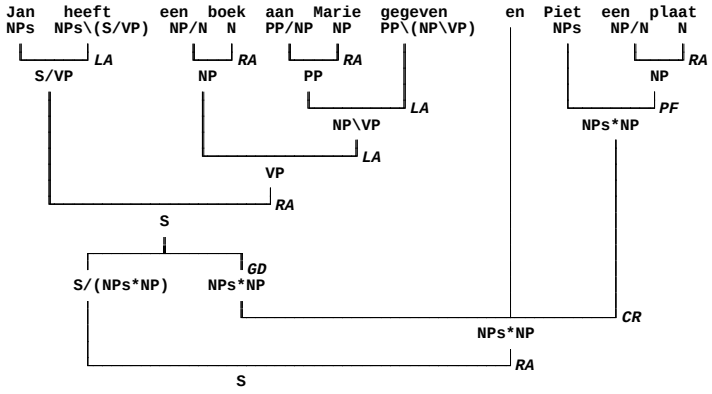
70.



‘Anatoly has two and Garry has three games won’
Anatoly has won two games and Garry has won three games

The constituency of expressions in a Gapping construction is as expected. The first conjunct is a full sentence, and the constituency facts are the same as when this sentence occurs isolated. The second conjunct, on the other hand, is put together by the product operator, and therefore lacks constituency. The whole sentence itself is naturally a constituent. In example (71), except for the lexical items and the whole sentence, the following word sequences are constituents: *Jan heeft*, *een boek*, *aan Marie*, *een plaat*, *aan Marie gegeven*, *een boek aan Marie gegeven*, *Jan heeft een boek aan Marie gegeven*.

71.



'Jan has a book to Marie given and Piet a record'
Jan has given a book to Marie and Piet a record

In the next section, at the end of chapter 3, I address Gapping phenomena, and the account of it in the PACG. Furthermore, it is interesting to examine how the combined Gapping and Right Node Raising cases would have to be accounted for. It appears to be necessary to extend the language specific rules on the basis of the linguistic material presented in section 3.5.

3.5 Extensions of the language specific rules

3.5.1 Introduction

In this section, I will discuss Gapping and combined Gapping / Right Node Raising examples. We will examine how these phenomena relate to a PACG-account. In section 3.5.2, I argue that the Gapping Decomposition rule should be restricted with some conditions. These conditions are captured in a Gapping Generalization. In section 3.5.3, we will discuss the account of Gapping, according to Morrill and Solias (1993), and finally in section 3.5.4, we will look at combined Gapping / Right Node Raising cases.

3.5.2 Gapping in the PACG

On the basis of Gapping phenomena in Dutch, I propose certain extensions of the language specific component of the PACG presented in section 3.2.

Gapping is a phenomenon of conjoining two sentences, the first one of which is a complete sentence and the second is one which lacks some parts. There are some conditions on the parts that are being left out (the gaps) and the parts that are present (the remnants). Neijt (1979) dedicates a quite thorough chapter on constraining the gaps and the remnants. In constraining the remnants of Gapping she concludes that remnants must be major constituents. In this she follows Hankamer (1973), who defines the notion of major constituent as follows:

> "A *major constituent* of a given sentence S_0 is a constituent either immediately dominated by S_0 or immediately dominated by VP, which is immediately dominated by S_0."
> Hankamer (1973: 18)

Although it is not self-evident that the frameworks of Phrase Structure Grammars and Categorial Grammars allow for similar generalizations, we could transform Hankamer's generalization into a CG-generalization of the following kind:

72. **Gapping Generalization**
Remnants of Gapping, i.e. the constituents of the second conjunct, have categories that equal the categories of the arguments of one of the verbs in the first conjunct.

The Gapping Generalization states that the constituents of the second sentence have the same categories as arguments of the verbs in the first conjunct, and in fact they act as arguments of the verbs. As we shall see, this is the case for Gapping in main clauses, and it implies that finite verb phrases cannot act as remnants of Gapping. But in subordinate clauses, we will see that even the finite verb can be a remnant of Gapping. The categorizations of the lexical items provided in chapter 1 indicate that verbs are either reducible to S or to VP (see definition 73).

73. **Definition of Verbs**
An expression φ is a verb if and only if it is an exocentric lexical item with a category reducible to S, or VP.

Definition (73) excludes verb modifiers from the class of verbs, because modifiers have an endocentric category. On the other hand, as we have seen in chapter 1, tensed verbs can be assigned a category VP_α/VP_β in subordinate clauses, with α=f (finite) and β=p (participle) or β=i (infinitival). These cases of verbs with different tense features will be considered exocentric.

To show the effect of the Gapping Generalization we present example (74), in which the verbs *heeft* (*to have*, with category NPs\(S/VPp)) and *gegeven* (*given*, with category PP\(NP\VPp)) and *teruggevraagd* (*asked back*, with the same category as *gegeven*) occur. The generalization predicts that only constituents with the categories NPs, VPp, PP or NP can occur as remnants of Gapping in the second conjunct. We see this in (74a) to (74i). Constituents that are arguments of one of these categories cannot be remnants in the second conjunct, as is shown in the sentences (74j) and (74k). In fact the possible remnants of Gapping must be supplemented with verbs, other than the finite verb. This is shown in (74l) to (74p). The categorial analyses of the first conjunct is presented in (75).

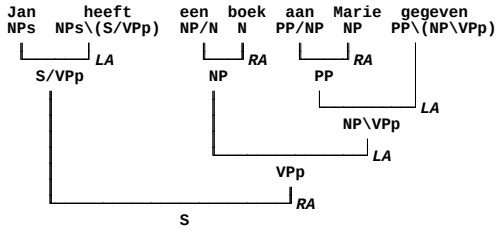
74. a [Jan]$_{NPs}$ heeft een boek aan Marie gegeven en [Piet]$_{NPs}$
b Jan heeft [een boek aan Marie gegeven]$_{VPp}$ en [een plaat aan Klaas teruggevraagd]$_{VPp}$
c. Jan heeft een boek [aan Marie]$_{PP}$ gegeven en [aan Klaas]$_{PP}$
d. Jan heeft [een boek]$_{NP}$ aan Marie gegeven en [een plaat]$_{NP}$
e. [Jan]$_{NPs}$ heeft [een boek aan Marie gegeven]$_{VPp}$ en [Piet]$_{NPs}$ [een plaat aan Klaas teruggevraagd]$_{VPp}$
f. [Jan]$_{NPs}$ heeft een boek [aan Marie]$_{PP}$ gegeven en [Piet]$_{NPs}$ [aan Klaas]$_{PP}$
g. [Jan]$_{NPs}$ heeft [een boek]$_{NP}$ aan Marie gegeven en [Piet]$_{NPs}$ [een plaat]$_{NP}$
h. [Jan]$_{NPs}$ heeft [een boek]$_{NP}$ [aan Marie]$_{PP}$ gegeven en [Piet]$_{NPs}$ [een plaat]$_{NP}$ [aan Klaas]$_{PP}$
i. Jan heeft [een boek]$_{NP}$ [aan Marie]$_{PP}$ gegeven en [een plaat]$_{NP}$ [aan Klaas]$_{PP}$

* j. Jan heeft een boek aan Marie gegeven en Piet [plaat]$_{N}$
* k. Jan heeft een boek aan Marie gegeven en Piet [Klaas]$_{NP}$

74. l. $[\text{Jan}]_{NPs}$ heeft een boek aan Marie $[\text{gegeven}]_{PP\backslash(NP\backslash VP)}$ en $[\text{Piet}]_{NPs}$ $[\text{terug-gevraagd}]_{PP\backslash(NP\backslash VP)}$
    m. Jan heeft $[\text{een boek}]_{NP}$ aan Marie $[\text{gegeven}]_{PP\backslash(NP\backslash VP)}$ en $[\text{een plaat}]_{NP}$ $[\text{terug-gevraagd}]_{PP\backslash(NP\backslash VP)}$
    n. $[\text{Jan}]_{NPs}$ heeft $[\text{een boek}]_{NP}$ aan Marie $[\text{gegeven}]_{PP\backslash(NP\backslash VP)}$ en $[\text{Piet}]_{NPs}$ $[\text{een plaat}]_{NP}$ $[\text{teruggevraagd}]_{PP\backslash(NP\backslash VP)}$
    o. $[\text{Jan}]_{NPs}$ heeft een boek $[\text{aan Marie}]_{PP}$ $[\text{gegeven}]_{PP\backslash(NP\backslash VP)}$ en $[\text{Piet}]_{NPs}$ $[\text{aan Klaas}]_{PP}$ $[\text{teruggevraagd}]_{PP\backslash(NP\backslash VP)}$
    p. $[\text{Jan}]_{NPs}$ heeft $[\text{een boek}]_{NP}$ $[\text{aan Marie}]_{PP}$ $[\text{gegeven}]_{PP\backslash(NP\backslash VP)}$ en $[\text{Piet}]_{NPs}$ $[\text{een plaat}]_{NP}$ $[\text{aan Klaas}]_{PP}$ $[\text{teruggevraagd}]_{PP\backslash(NP\backslash VP)}$

    'Jan has a book to Marie given and (Piet) (a record) (to Klaas) (asked back)'
    Jan has given a book to Marie and Piet has asked back a record from Klaas

75.



(74b) Can either be analysed as ordinary coordination of VP_p, or, according to the definition in (72), as Gapping. Note that in (74k) the NP in the second conjunct is an argument of the functor *aan* (with category PP/NP), and not of the verb *gegeven* (with category PP\(NP\VPp)). In the latter case, assuming that we could give people away, the sentence becomes grammatical, just like sentence (74g).

In the light of sentence modifiers or VP modifiers, the generalizations need some adjustment. The sentences in (76) and (77) show that such modifiers can also function as remnants in the second conjunct.

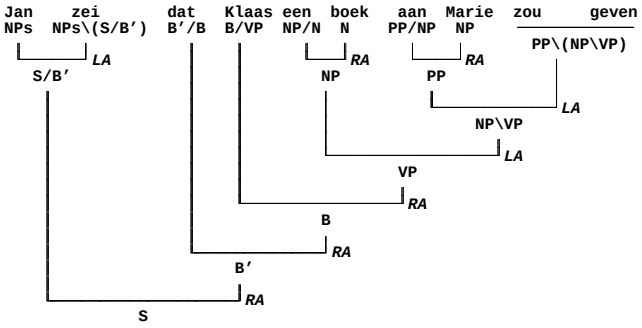
76. Jan heeft $[\text{vandaag}]_{VPp/VPp}$ een boek $[\text{aan Marie}]_{PP}$ gegeven en $[\text{gisteren}]_{VPp/VPp}$ $[\text{aan Klaas}]_{PP}$
    'Jan has today a book to Marie given and yesterday to Klaas'
    Jan has given a book to Marie today and to Klaas yesterday

77. [Waarschijnlijk]$_{S/S}$ heeft Jan [een boek]$_{NP}$ [aan Marie]$_{PP}$ gegeven en [vermoedelijk]$_{S/S}$ [een plaat]$_{NP}$ [aan Klaas]$_{PP}$
'Probably has Jan a book to Marie given and presumably a record to Klaas'
Jan has probably given a book to Marie and presumably a record to Klaas

Unlike finite verb phrases in main clauses, the finite verb phrases in subordinate clauses can function as remnants of Gapping. In example (78), I present two instances of finite verb phrases as remnants. In (79), the analysis of the first part of the sentence is given. In example (80), we can observe that all remnants must be part of the same clause level.

78. a. Jan zei dat Klaas een boek aan Marie zou geven en Kees zou terugvragen
'Jan said that Klaas a book to Marie would give and Kees would ask back'
Jan said that Klaas would give a book to Marie and that Kees would ask back a book from Marie
b. Jan zei dat Klaas een boek aan Marie zou geven en een plaat zou terugvragen
'Jan said that Klaas a book to Marie would give and a record would ask back'
Jan said that Klaas would give a book to Marie and would ask back a record

79.



80. a. * Jan zei dat Klaas een boek aan Marie zou geven en [$_{NPs}$Piet] [$_{NP}$Kees] [$_{NP}$een plaat]
b. * Jan zei dat Klaas een boek aan Marie zou geven en [$_{NPs}$Piet] [$_{NP}$een plaat] [$_{PP}$aan Karin] [$_{PP\backslash(NP\backslash VP)}$zou terugvragen]

The above observations lead to an adjustment of the Gapping Generalization (72), concerning possible remnants. The adjustment is presented in (81). The notion *clause*, used in Gapping Generalization (81), is defined in definition (82).

81. **Gapping Generalization** (adjustment)
    a. All remnants are constituents of the same clause;
    b. In main clauses, remnants are arguments of the verbs present in the first conjunct, functions over the verbs or the verbs themselves (except from the finite verb of the matrix clause);
    c. In subordinate clauses, remnants are arguments of the verbs, functions over the verbs or the verbs themselves.

82. **Clause**
    Two constituents, X and Y, belong to the same *clause* -S, B or Q- iff the first clause that is reached after reducing X with its functions and arguments is the same as for Y.

Furthermore definition (81) states what elements are necessarily and what elements are possibly gapped. In the first place Gapping is a phenomenon which leaves out the tensed verb of the matrix clause of the second conjunct. This constituent is necessarily left out in the second conjunct. This is expressed in (81.b) by stating that the finite verb is not a possible remnant. For the possible gaps we need no separate generalization. The possible gaps are defined by the mirror image of the statements b and c of the definition on remnants (81).

The Gapping Generalization appears to account for ungrammaticalities which need constraints based on the Tensed S Condition in Phrase Structure Grammars. The Tensed S Condition is stated as follows:

> **Tensed S Condition**
> No rule can involve X, Y in the structure ... X ... [$_{\alpha}$ ... Y ...] ...
> where Y is not in COMP and α is a tensed sentence.
> Neijt (1979: 141) (quoting Chomsky (1973: 244))

Neijt furthermore states:

> "For Gapping, the Tensed S Condition claims that tensed sentences cannot contain one of the remnants but not the other, unless the remnant contained in the tensed sentence is in COMP."
> Neijt (1979: 142)

This accounts for the differences in the sentences (83) and (84). In sentence (83), the remnant *welke meisjes* (*which girls*) is in COMP, where in sentence (84) the remnant *dertig meisjes* (*thirty girls*) is not in COMP.

83. Karel mag beslissen welke jongens er mee gaan en Max welke meisjes
Karel may decide which boys are coming along and Max which girls

84. * Karel besliste dat er twintig jongens mee zouden gaan en Harrie dertig meisjes
Karel decided that twenty boys would come along and Harrie thirty girls

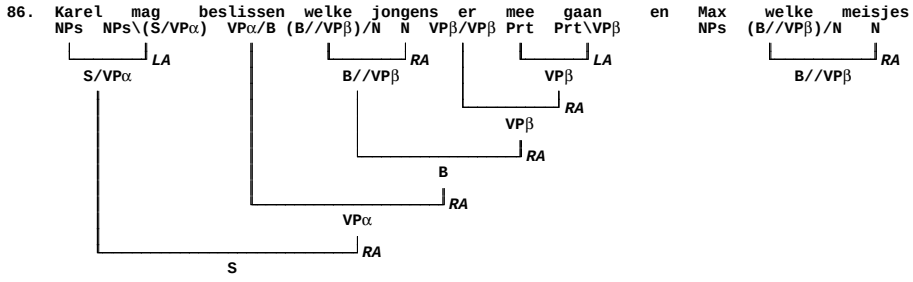
Assuming the categorizations in (85), we immediately see that sentence (84) is not recognized by the categorial grammar outlined in this chapter, because the only possible remnants in (84) are the subject NP (NPs) and the subordinate clause, including the subordinator (B'). For the grammatical sentence (83), there seems to be a problem. When *beslissen* is categorized as VP/B, then the only possible remnant this category calls for is B. The fact is that we do want to be able to treat the NP *welke meisjes* as a possible remnant too. What we need then, is a way of treating the word sequence *welke meisjes* as a B for this purpose. I do so by assigning the category (B//VP)/N to *welke* in the lexicon. Together with *meisjes*, *welke* yields the constituent *welke meisjes* with category B//VP. X//Y is different from X/Y in that it functions as an X. I suppose the combinatory rules for X//Y to be the same as for X/Y. The double slash has been used before (see for instance Bouma (1986) to account for extraction phenomena), but the way I use the double slash is just to mark a category X//Y to be able to act as an X in certain circumstances. In this sense, it more resembles for example Gazdar's slashed categories X/Y (Gazdar (1981)). These were meant to indicate a Y hole in a category X. *Beslissen* in sentence (83) must be assigned the category VP/B. We must now think of *welke meisjes* as the argument of *beslissen*, although the argument itself is an unsaturated function over sequences of the category VP, of

which *er mee gaan* is an instance. The (un)grammaticality of (86) and (87) is now accounted for by the Gapping Generalization (81).

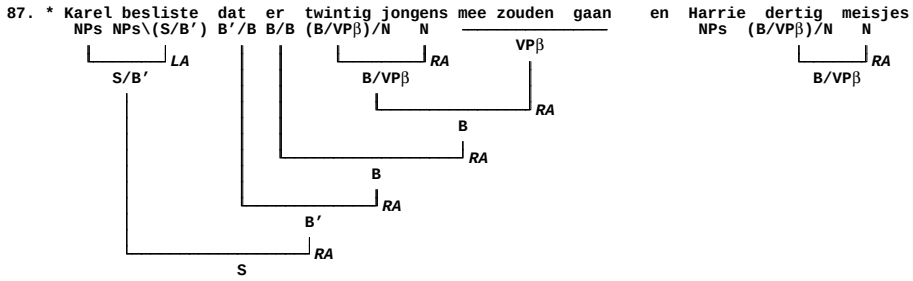
85.

	Karel, Max, Harrie	→ NPs
	mag	→ NPs\(S/VP_α)
	besliste	→ NPs\(S/B')
	dat	→ B'/B
	er	→ B/B, VP_β/VP_β
	twintig, dertig	→ (B/VP_β)/N
	jongens, meisjes	→ N
	mee	→ Prt
	zouden	→ VP_β/VP_β
	gaan	→ Prt\VP_β
	beslissen	→ VP_α/B
	welke	→ (B//VP_β)/N

86.



87.



The approach sketched in this section not only accounts for the grammaticality of (86) and the ungrammaticality of (87), but also accounts for the grammaticality judgements concerning the examples in (88).

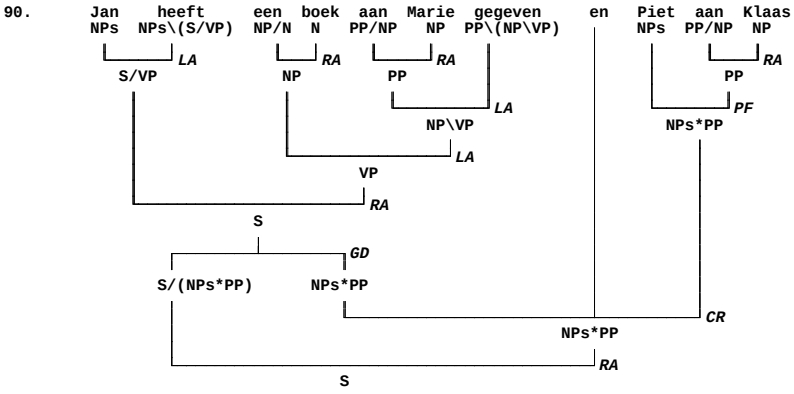
88. a. * Karel mag beslissen welke jongens er mee gaan en Max er thuisblijven
Karel may decide which boys are coming along and Max are staying home
b. Karel mag beslissen welke jongens er mee gaan en Max welke jongens er thuisblijven
Karel may decide which boys are coming along and Max which boys are staying home
c. Karel besliste dat er twintig jongens mee zouden gaan en Harrie dat er dertig meisjes thuis zouden blijven
Karel decided that twenty boys would come along and Harrie that thirty girls would stay home

Up to now in this section we only gave a full analysis of the left conjunct of gapped sentences. Before we turn to the full analysis of the right conjunct, and of the whole sentence as well, we should first pay attention to the nature of Gapping. Gapping can be considered to be a parallel coordination. The categories of the constituents that are being left out in the second conjunct equal their counterparts in the first conjunct. Since in categorial grammar there is no such thing as deletion, we are forced to reorder the categories of the first conjunct (an alternative analysis by Morrill and Solias (Morrill and Solias (1993)) will be presented in section 3.5.3). In doing this we will follow Steedman (1990) to a large extent. The Gapping Decomposition Rule, introduced in section 3.2.4 will, however, differ from Steedman's Decomposition Rule in the use of strings as product categories. We repeat the Dutch Gapping Decomposition Rule (24) here as (89) and we extend it with a condition on X_1 to X_n.

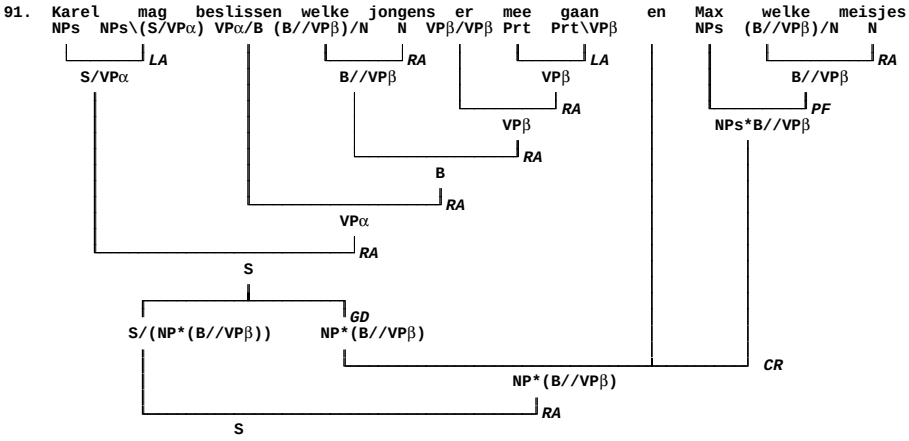
89. **Gapping Decomposition**
S $\Rightarrow$ S/X X, where X = X_1*...*X_n and X_1...X_n satisfy Gapping Generalization (81)

As examples of Gapping we will give the derivations of (74f) in (90) and of (85) in (91).

90.



91.



3.5.3 Gapping according to Morrill and Solias

In Morrill and Solias (1993), Gapping (among other phenomena) is treated in a Lambek/Gentzen-style categorial grammar. This kind of grammar functions as a logical deduction system, where axioms and inference rules decide what sequences can be deduced from other sequences. The logic of the system is independent of the natural languages they describe. The differences between natural languages are captured by defining different operators in the grammar for these languages.

In this way Morrill and Solias treat five phenomena by means of three non-standard operators (regarding the division and the product symbol as standard

operators). The phenomena are: particle-verb constructions (92), discontinuous idioms (93), sentential scope of the quantifier (94), pied piping (95), and Gapping (96). These phenomena have in common that some kind of discontinuity has to be accounted for. The three non-standard operators used for this account are *wrapper*, *infixer* and the *discontinuous product*.

92. Mary **rang** John **up**

93. Mary **gave** John **the cold shoulder**

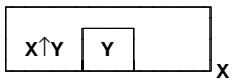
94. John likes everything

95. for whom John works

96. John studies logic, and Charles phonetics

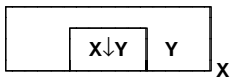
Wrapper is symbolised by $\uparrow$, and it accounts for functors $X\uparrow Y$ wrapping around their arguments Y to reduce to X. Schematically it can be illustrated by the following picture (figure 1).

Figure 1: wrapper



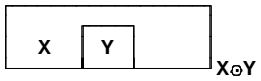
Infixer is symbolised by $\downarrow$, and it accounts for functors $X\downarrow Y$ putting themselves inside their arguments Y to reduce to X. This can be illustrated by figure 2.

Figure 2: infixer

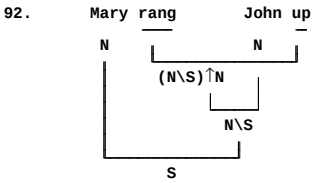


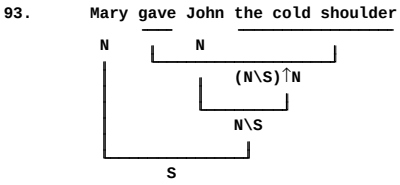
The *discontinuous product* $X\odot Y$ accounts for products of non adjacent categories. The Y must be infixed in X. The picture that illustrates this, is figure 3.

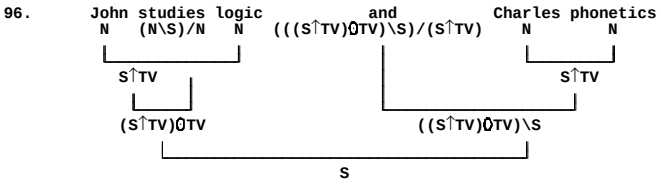
Figure 3: discontinuous product



To show how the three new operators account for the above sentences I give the derivation of the sentences (92), (93), and (96).







In sentence (92) the particle-verb construction *rang up* has the category (N\S)↑N. It wraps around *John* to form the constituent *rang John up*, which has the category N\S. Applying the constituent *Mary*, N, to this function yields the sentence *Mary rang John up*.

Sentence (93) shows the same reduction schemes as (92). *Gave the cold shoulder* functions as *rang up*.

Example (96) shows how Gapping is treated. The extremely complex category for *and* accounts for application of the two nouns in the second conjunct, and consecutive application of the first conjunct. The nouns in the first and second conjunct are composed to the category S↑TV, where TV = (N\S)/N. It is obvious

that the two nouns should wrap around the transitive verb, and the Lambek/Gentzen theorems for these operations are easy to conceive. But in a traditional derivational categorial grammar, like the one outlined in this chapter, it is hard to reach the category S↑TV by type lifting and functional composition. It seems that for discontinuous constituents other mechanisms than the tree building ones are required. Furthermore, not all Gapping cases concern just one missing transitive verb. Morrill and Solias do not show how more complex cases, like the ones presented in the previous section, are accounted for. If you look at sentence (90) you see that there is more to Gapping than two noun phrases wrapping around a transitive verb, because not only the remnants, but also the gapped constituents form discontinuous sequences.[47]

A further question arises considering the discontinuous product. This operator accounts for the combination of *John logic* with *studies* to an expression of category (S↑TV)⊙TV, but when and where is the discontinuous product called for? Does the grammar develop for every tuple of 'wrapping expression/ argument' such a discontinuous product category? In other words: Why is the discontinuity property encoded twice? Wouldn't (S↑TV)*TV do the job, or (S/TV)⊙TV?

The sentences (94) and (95) are more problematic from a categorial point of view. The categories that are assigned to the constituents do not seem to account for the word orders at issue.

94. **John likes everything**
N (N\S)/N S↓(S↑N)

95. **for whom John works**
PP/N (R/(S↑PP))↓(PP↑N) N (N\S)/PP

---

[47] In the yet unpublished paper 'Unassociative Tuple, Sequences, Discontinuity and Gapping in Categorial Grammar' by Teresa Solias (1993) generalized discontinuity operators are proposed to account for multiple gaps. Solias says: "Nevertheless, not only is this type (the type for *and* in Morrill and Solias (1993), JH) rather complex and hardly intuitive but it cannot either account for iterative gapped sentences." (Solias 1993: 29). She proposes two new operators to account for (multiple) gapping, the operator ♦ which is a product type with range over the present lexical items, and the inclusion operator **o**, which apparently makes it possible to recover the gapped material repeatedly. In the end she proposes the type (S**o**(S↓(A♦B))\S)/(A♦B) for the conjunction.

It is hard to see where *everything* in sentence (94) is being infixed in, and what sequence has the category S↑N. *John likes* is an expression of category S/N, but Morrill and Solias fail to explain how this category is changed in S↑N. In example (95), *whom* is apparently being infixed in a PP↑N, a category that is not present, and if it were present it would probably be the singleton constituent *for*. For the same reason it remains unclear how the category of *John works*, S/PP, is changed into S↑PP.

On the one hand it looks like Morrill and Solias need quite some gymnastics to account for Gapping phenomena. The infixing and wrapping mechanisms, along with the discontinuous product, can account for Gapping within the logical deduction systems, without disturbing the logic of the system itself. But the results of the new operators are that infixing does not mean infixing all the time, and the same holds m.m. for wrapping.

On the other hand questions arise as to how the new operators work in the sentential quantification and the pied piping cases. Furthermore this treatment does not account for the intuition expressed in this study that the connective *and* combines two parallel structures.

The consequences of the treatment in 3.2.4 are that the conjunction can function as operating on parallel structures, which is a desirable feature. On the other hand, the PACG of section 3.2.3 and 3.2.4 inevitably contains a decomposition mechanism in order to accomplish the parallel structures.

3.5.4 Gapping and Right Node Raising

The decomposition analysis in section 3.2.4 assumes a complete S as the left conjunct. In the usual gapped sentences this is the case, but there are cases in which Gapping and Right Node Raising co-occur in a sentence. The grammar, outlined so far, doesn't cover these cases. I will make some suggestions for an account of this phenomenon, compatible with the PACG.

In combined Gapping and Right Node Raising cases the left conjunct does not consist of a complete S. In the examples (96) to (121) a wide range of conjoined

Gapping / Right Node Raising cases are presented. The gapped parts are in italics and the Right Node Raising parts are in bold type. The glosses and translations will not be given for each sentence but group wise. The translations within the groups correspond to each other.

96. Barbara *heeft* het zure en Simon het zoete **snoepje gegeten**
97. Barbara *heeft gisteren* het zure en Simon het zoete **snoepje gegeten**
'Barbara has (yesterday) the sour and Simon the sweet candy eaten'
Barbara has eaten the sour candy (yesterday) and Simon has eaten the sweet candy (yesterday)

98. * Barbara *heeft een knikker* en Simon **gevonden**
99. * Barbara *heeft* gisteren *een knikker* en Simon vandaag **gevonden**
'Barbara has (yesterday) a marble and Simon (today) found'
Barbara has found a marble (yesterday) and Simon has found a marble (today)

100. * Jan *heeft een boek aan de blinde* en Piet **gegeven**
101. * Jan *heeft* gisteren *een boek aan de blinde* en Piet vandaag **gegeven**
102. * Jan *heeft een boek* en Piet **aan de blinde gegeven**
103. * Jan *heeft* gisteren *een boek* en Piet vandaag **aan de blinde gegeven**
104. * Jan *heeft* en Piet **een boek aan de blinde gegeven**
105. Jan *heeft* gisteren en Piet vandaag **een boek aan de blinde gegeven**
'Jan has (yesterday) a book to the blind and Piet (today) given'
Jan has given a book to the blind (yesterday) and Piet has given a book to the blind (today)

106. * Jan *heeft* een boek *aan de blinde* en Piet een plaat **gegeven**
107. * Jan *heeft* gisteren een boek *aan de blinde* en Piet vandaag een plaat **gegeven**
108. Jan *heeft* een boek en Piet een plaat **aan de blinde gegeven**
109. Jan *heeft* gisteren een boek en Piet vandaag een plaat **aan de blinde gegeven**
'Jan has (yesterday) a book to the blind and Piet (today) a record given'
Jan has given a book to the blind (yesterday) and Piet has given a record to the blind (today)

110. Jan *heeft een boek* aan de blinde en Piet aan de dove **gegeven**
111. * Jan *heeft* gisteren *een boek* aan de blinde en Piet vandaag aan de dove **gegeven**
'Jan has (yesterday) a book to the blind and Piet (today) to the deaf given'
Jan has given a book to the blind (yesterday) and Piet has given a book to the deaf (today)

112. Jan *heeft* een boek aan de blinde en Piet een plaat aan de dove **gegeven**
113. Jan *heeft* gisteren een boek aan de blinde en Piet vandaag een plaat aan de dove **gegeven**
'Jan has (yesterday) a book to the blind and Piet (today) a record to the deaf given'
Jan has given a book to the blind (yesterday) and Piet has given a record to the deaf (today)

114. *Jan heeft* een boek en een plaat **aan de blinde gegeven**
115. *Jan heeft* gisteren een boek en vandaag een plaat **aan de blinde gegeven**
116. * *Jan heeft* een boek *aan de blinde* en een plaat **gegeven**
117. * *Jan heeft* gisteren een boek *aan de blinde* en vandaag een plaat **gegeven**
'Jan has (yesterday) a book and (today) a record to the blind given'
Jan has given a book to the blind (yesterday) and Jan has given a record to the blind today)

118. *Jan heeft* een boek aan de blinde en een plaat aan de dove **gegeven**
119. *Jan heeft* gisteren een boek aan de blinde en vandaag een plaat aan de dove **gegeven**
'Jan has (yesterday) a book to the blind and (today) a record to the deaf given'
Jan has given a book to the blind (yesterday) and Jan has given a record to the deaf (today)

120. *Jan heeft een boek* aan de blinde en aan de dove **gegeven**
121. *Jan heeft* gisteren *een boek* aan de blinde en vandaag aan de dove **gegeven**
'Jan has (yesterday) a book to the blind and (today) to the deaf given'
Jan has given a book to the blind (yesterday) and Jan has given a book to the deaf (today)

The above examples show that there is less liberty in combined Gapping / Right Node Raising, than there is in Gapping. In the combined Gapping / RNR-cases, the Gapping concerns a linear successive sequence of constituents in the first con-

junct, otherwise the result is ungrammatical. This linearity of the gapped constituents is a necessary but not a sufficient condition.

The regularities deduced from the examples above are hard to account for in a formal way in the PACG. We will, nevertheless, try to provide a corollary of these regularities in (122). Because of the lack of formal robustness, we are not able to give this corollary the status of rule in the PACG.

The constituents in the first conjunct, to be gapped in the second conjunct, are being denoted by Y_1 to Y_n in the Gapping Permutation Regularity (GP) (122). This corollary permutes the words in the first conjunct in a way that makes coordination with the second conjunct possible. The condition on the linear succession of the gapped constituents rules out the sentences (99), (101), (103), (106), (107), (111), (116), and (117).

The examples (98), (100), (102), and (104) all exhibit linear succession in the first conjunct of the constituents that are gapped in the second conjunct. Nevertheless, these sentences are all ungrammatical. They have one characteristic feature in common, namely that there is no linguistic material between the sequence in the first conjunct, that is gapped in the second conjunct, and the coordinator. In the grammatical examples there always is linguistic material in between these two. We can capture this fact in a condition on the Gapping Permutation Regularity (122), namely the condition that $0<\beta$. In other words, there is at least one Z_β right to the gapped expressions.

122. **Gapping Permutation Regularity (GP)**

a. $X\ V_t\ Y_1...Y_n\ Z_1...Z_m$ **en** $\Rightarrow$ $V_t\ Y_1...Y_n\ X\ Z_1...Z_m$ **en**
The variables X, Y_α, Z_β, are arguments of one of the verbs, functions over one of the verbs, or verbs themselves, where $0<\beta\leq m$. The variable Z_m may be a part of such a constituent.

b. If $X\ V_t\ Y_1...Y_n\ Z_1...Z_m$ $\Rightarrow$ $V_t\ Y_1...Y_n\ X\ Z_1...Z_m$,
for some X, $Y_1...Y_n$, $Z_1...Z_m$
then the inverse inference
$V_t\ Y_1...Y_n\ X(*)Z_1(*)...(*)Z_m$ $\Rightarrow$ $X\ V_t\ Y_1...Y_n\ Z_1(*)...(*)Z_m$ must be derived in the remainder of the derivation.

As an illustration we present a few derivations. In the sentences (123) and (124), the above (104) and (105), the condition $0<\beta$ of the Gapping Permutation

Regularity plays a decisive role. In the examples (125) and (126), the above (115) and (116), the linear succession of the gapped constituents accounts for the (un)grammaticality.

123. * Jan heeft en Piet een boek aan de blinde gegeven
NPs NPs\(S/VPp) NPs NP/N N PP/NP NP/N N PP\(NP\VPp)
GP
NPs\(S/VPp) NPs (on the ground of the condition 0<β)

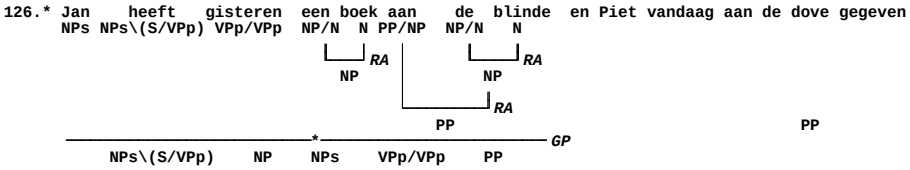
‘Jan has and Piet a book to the blind given’
Jan has given a book to the blind and Piet has given a book to the blind

124. Jan heeft gisteren en Piet vandaag een boek aan de blinde gegeven
NPs NPs\(S/VPp) VPp/VPp NPs VPp/VPp NP/N N PP/NP NP/N N PP\(NP\VPp)
GP
NPs\(S/VPp) NPs
PF
NPs*(VPp/VPp)
PF
NPs*(VPp/VPp)
RA
NP
RA
NP
CR
NPs*(VPp/VPp)
RA
PP
GP
NPs NPs\(S/VPp) VPp/VPp
LA
NP\VPp
LA
VPp
LA
S/VPp
RA
VPp
RA
S

‘Jan has yesterday and Piet today a book to the blind given’
Jan has given a book to the blind yesterday and Piet has given a book to the blind today

125. Jan heeft een boek aan de blinde en Piet aan de dove gegeven
NPs NPs\(S/VPp) NP/N N PP/NP NP/N N NPs PP/NP NP/N N PP\(NP\VPp)
RA
NP
RA
NP
RA
NP
RA
PP
RA
PP
GP
NPs\(S/VPp) NP NPs PP
PF
NPs*PP
PF
NPs*PP
CR
NPs*PP
GP
NPs NPs\(S/VPp) NP PP
LA
S/VPp
LA
NP\VPp
LA
VPp
RA
S

‘Jan has a book to the blind and Piet to the deaf given’
John has given a book to the blind and Piet has given a book to the deaf



'Jan has yesterday a book to the blind and Piet today to the deaf given'
Jan has given a book to the blind yesterday and Piet has given a book to the deaf today

In the next section, 3.6, we present the conclusions of this third chapter.

3.6 Concluding remarks

In this chapter, we argued that coordination phenomena can be accounted for straightforwardly in a product-based applicative categorial grammar. The core part of this grammar is the parentheses-free AB-grammar, named after Ajdukiewicz (1935) and Bar-Hillel, Gaifman and Shamir (1960), supplemented with the product operator. One important feature of the product operator, as it is used in this thesis, is its associativity. The universal grammar that we build up this way, is structurally complete. This is in no way threatening to the grammar, or to the languages described by the grammar. On reason for this is that reduction only takes place by means of the application rules. There are no other reductions possible or provable, than the application rules. The second reason the structural completeness is harmless with respect to the structures that are recognized is the fact that constituency is not lost in this approach, contrary to the approach of the Lambek-calculus. By defining expressions to be constituents if and only if their categories are product-free, we restrict constituency to applicative structures.

The language specific part of the grammar for Dutch, consists of three rules. The most important one, from the point of view of this thesis, is the Coordination Rule. The Coordination Rule accounts for parallel coordination phenomena. As we discussed in chapter 1, there is a wide variety of asymmetrical coordination.

We have left them out of discussion in chapter 3. The reason I did this, is that most coordinations obey the parallelism requirement, even Gapping.

The second language specific rule, presented in chapter 3, is the Disharmonic Composition Rule. We discussed the disharmonic structuring of the Dutch verb cluster. In this discussion, we concluded that even when we account for verb clustering in the lexicon, we need a syntactic rule of Disharmonic Composition. This Disharmonic Composition Rule is regarded as a reductional rule, and as a consequence there also exists a product version of this rule, the Product Disharmonic Composition.

The third language specific rule we discussed is the Gapping Decomposition Rule. This rule reanalyzes the first part of a Gapping coordination. This first part is a sentence which can be reanalyzed as a function over possible remnants, followed by the remnants. After the Decomposition, the Coordination Rule can apply. In this way, we also treat Gapping as a parallel coordination. In this respect, our account of Gapping is different from the account Morrill and Solias (1993) worked out.

The language specific rules all have in common that they can be restricted to a certain class of expressions. In section 3.2.5, we restricted the Coordination Rule. The restriction I formulated, defined that the head of an NP or a PP must be in the coordination, if its arguments (dependents) are part of the coordination.

The Disharmonic Composition Rule and the Product Disharmonic Composition are also restricted. As the rule is supposed to account for the verb clustering in Dutch, only main functors over VP's should be allowed to enter these rules.

The Gapping Decomposition Rule must be restricted to certain classes of gaps and remnants. It appeared to be hard to formulate these restriction on functions and arguments. Some more general statements on Gapping, and the possible remnants of Gapping, had to be proposed. These are captured in the Gapping Generalization in section 3.5.2.

In section 3.3, we discussed Mary Wood's *Thursday Grammar*. She was the first to make an explicit use of product categories in categorial grammar. Wood's

grammar and my PACG have some things in common. One thing is the Product Formation Rule, of course. But, other than my starting point, she admitted harmonic forms of the composition rules. The reason for Wood to introduce product rules, was to avoid Generalized Composition, an operation which makes it possible to combine a main functor looking for a Y, and a subordinate functor with an embedded Y in numerator. The Product Formation Rule was used to present the arguments of the Y in the subordinate clause as a flat structure, after which the main and subordinate functor could be composed by simple Composition. In the PACG, however, both the Harmonic Composition Rules and the Type Lifting Rules are excluded from the grammar.

Finally, I will present the PACG in its ultimate form.

127. **Product-based Applicative Categorial Grammar (PACG)**

Universal Rules

Reduction Rules

Right Application (RA)
X/Y Y ⇒ X
Left Application (LA)
Y Y\X ⇒ X

Product Rules

Product Formation (PF)
X Y ⇒ X*Y
Product Associativity (PA)
(X*Y)*Z ⇔ X*(Y*Z)

Derived Reduction Rules

Right Product Application (RP)
Γ*((X/Y)*Y)*Δ ⇒ Γ*X*Δ
Left Product Application (LP)
Γ*(Y*(Y\X))*Δ ⇒ Γ*X*Δ

127. **Product-based Applicative Categorial Grammar (PACG)**

Dutch Rules

Coordination Rule (CR)

X^{+} **en** X $\Rightarrow$ X, where X ≠ N*NP\Y and X ≠ NP*PP\Z

Disharmonic Rules

Disharmonic Composition (DC)

X/VP $Y_i\backslash(...\backslash(Y_n\backslash VP))$ $\Rightarrow$ $Y_i\backslash(...\backslash(Y_n\backslash X))$, where X≠B

Product Disharmonic Composition (PD)

$\Gamma*(X/VP*Y_i\backslash(...\backslash(Y_n\backslash VP)))*\Delta$ $\Rightarrow$ $\Gamma*Y_i\backslash(...\backslash(Y_n\backslash X))*\Delta$, where X≠B

Gapping Decomposition (GD)

S $\Rightarrow$ S/X X, where X = $X_1,...,X_n$ and $X_1,...,X_n$ satisfy the Gapping Generalization

Gapping Generalization

a. All remnants are constituents of the same clause;
b. In main clauses, remnants are arguments of the verbs present in the first conjunct, functions over the verbs or the verbs themselves (except from the finite verb of the matrix clause);
c. In subordinate clauses, remnants are arguments of the verbs, functions over the verbs or the verbs themselves.