1 On the Categorial Characterization of Natural Languages

1. On the categorial calculi

1.1. Grammar and grammar

The notion of grammar is two-sided. On the one hand, a grammar may be a set of material statements about certain languages, languages in general or even language-as-such. These statements express certain intrinsic properties of the linguistic object in question. On the other hand, the notion of grammar may refer to a set of formal or informal instruments, to be used in descriptive or theoretical linguistics.

Janssen (1983)'s title, `Foundations and Applications of Montague Grammar', is almost programmatic in stressing the second, instrumental use of the notion. There is, as we may infer from it, grammar before its being applied to natural and formal languages, and there may even be grammar that is not applied at all. But, as we might have studies into the foundations of generative grammar (`generative' taken ideologically — not, as in the introduction to Gazdar and others 1985, technically) it would make no sense to write or read studies in applied generative grammar. One might be blamed for missing the point.

Linguists and linguistic schools may differ in their commitment to the one or other exegesis of the art of doing grammar. In the generative corner, a strong tendency towards the material interpretation of grammar can be observed. At least since the rise of the government-and-binding type of theories, generative linguists are concerned more with *what* should be said of natural language rather than with *how* these statements are to be made. They are, as I see it, not particulary interested in exploiting a fine and small selection of tools. Whenever certain means of expression are changed, added, rejected or neglected, this is not due to instrumental considerations but to their content. Generative grammar is driven by relevance and hypothesis, not by instruments and logic. As a consequence, methodology is somewhat underdeveloped in this area of modern linguistics. Chomsky's suggestion that a lack of interest in methodological issues characterises full fledged science (Chomsky 1988: 190), is misguided, however.

Bach (1988) distinguishes between linguistic theories by parametrizing semantic accountability and generativity. As to the latter dimension, Bach refers to theories

for which "(t)he primary work goes into stating constraints or licenses" as *constitutional*, as opposed to *architectural* approaches, in which "the attempt is made to build into the very structure of a grammar various properties from which will follow principles stipulated independently in the alternative sort of theory" (p. 19). This distinction is appealing because it stresses the way principles of language are generated: by reflection and hypothesis driven inquiry in constitutional theories, by building and testing grammars in architectural theories. The attack on language is direct in constitutional, indirect in architectural theories. Though it is noteworthy that Bach's division is along strategic, rather than ideological dimensions, the main question it raises is whether the set of principles stated in the one theory turns out to be equivalent to the set of principles derived in the other.

There is a direct line from here to the types of relations between parsers and grammars Fodor (1985) introduces. She distinguishes parsers that can just be made to incorporate or obey some grammatical constraint from parsers that run more efficiently under this constraint and from parsers that entail the constraint in some explanatorily satisfying way. Van de Koot (1990) argues that the actual relation between a parser and a linguistic constraint may be hard to detect, but the issue is clear: a parser can be a dedicated automation of some particular (concept of a) grammar or just an instance of a very general way of running machines applied to a grammar, or something in between. In the parsing of natural language too, architectural and constitutional tendencies may struggle for precedence: is the parser to be defined by the grammar or just fed by it? And if there happens to be a non-trivial connection between the function computed by the parser and the function embodied by the grammar (in Van de Koot's terms), should this connection result from brute restriction or from deep affinity?

The study of grammar would not be a science, as it is claimed to be — a very stubborn branch according to Shieber (1988) — if in daily practice the alleged dichotomy of (constitutional) materialism and (architectural) instrumentalism were that strong. Nevertheless, many of the actual overt and covered discrepancies among grammarians, diverging from disabled communication to disrespect of each others work, seem to root in the one's judgement about the other being concerned too much or too little with one aspect of grammar. The overall lack of enthusiasm for the ambitious `deterministic enterprise' of Marcus (1980), for example, is easily explained along this line. Marcus defined a (run-time) `Item-and-Process' grammar, Parsifal, that hardly abounds in elegance and straight design. Therefore, Parsifal was not welcomed by instrument-oriented grammarians, the architects. But in actually writing and parsing a grammar in a principled way, Marcus had to do a lot of bookkeeping and to go deep into engineering more or less trivial phenomena that obscure, within the chosen setting, more fundamental aspects of the grammar that he aimed to parse. For that, Marcus' model was not welcomed among the material grammarians either. Perhaps, Van

de Koot (1990) contributes to the rehabilitation of the Parsifal mix: engineering grammar.

1.2. The categorial toolbox

Categorial grammar springs off the instrumental side. It can hardly be seen as a dedicated theory of natural language. In fact, categorial grammars are multi-purpose systems, equally well suited for electronic engineering, knowledge representation and the description and computation of languages — a property they share with rewriting grammars, by the way. Categorial grammars, by their very nature, rephrase whatever system they describe as a language, i.e. as one-dimensional concatenation. Using categorial grammar in the description of natural language amounts to putting back natural languages into the gallery of ordinary systems. To the linguist it offers a well-defined notion of category as a class of combinatorially equivalent terms (see e.g. Buszkowski 1988a) and a well-defined notion of combinatory rule. Basically, it is easy to `grammar' a language categorially. All one has to do is to define a set of categories, to label the lexicon accordingly and to select some rules. The coherence of the resulting framework relates to the idea of having a calculus that may impose constraints on the set of possible statements and enables a comparison of your choice with other viable selections of categories and rules.

Yet, there is a complementary perspective for categorial grammar. Sommers (1982: Introduction) distinguishes a Fregean from an Aristotelian position as to the logic of natural language. Frege qualifies as a `constructionist' because of his conviction that the logic of syntax equals the syntax of some artificial logic, whereas Aristotle is of the `naturalist' persuasion, preaching that syntactic structure is `in close correspondence' to logical form. Under a scalar interpretation of Sommers' dichotomy, Montague will be more of a constructionist than Chomsky, and approaches to natural language within the framework of generalized quantifiers are utterly naturalistic, witness Barwise and Cooper (1981) and, in particular, Zwarts (1986). Full categorial grammar, taking — unlike Montague (1973) — a calculus as the engine of syntax, is to be situated near the naturalist pole, closer to e.g. the Logical Form branch of generative grammar than to Fregean linguistics. Sketchy though these scalars may be, they suggest that the space for theories of grammar is multidimensional and that the quest for mainstream may yield a meander rather than a canal.

By now, categorial calculi come in great variety. Theorems of the calculi are of the form

(1) $\mathbf{x_1,\ldots,x_n \rightarrow y}$

which reads as: the sequence of categorial types $\mathbf{x}_i$ reduces to the categorial type **y**, or equivalently: the sequence of categorial types $\mathbf{x}_i$ is of type **y**. For any given set of types TYPE, the calculi are partially ordered by the subset relation on the sets of theorems over TYPE that they license. A calculus C whose set of theorems over TYPE is properly included in the set of theorems licensed by C′, is said to be weaker than C′.

In order to compare calculi, it is useful to have the set TYPE fixed by some recursive definition. Throughout this and following chapters, TYPE will be considered to arise from a non-empty set of primitive and atomic types BASE and a couple of operators {/, \} by the following recursion.

(2) (a) BASE is a proper subset of TYPE.
(b) If **a** and **b** are in TYPE, **a/b** and **a\b** are in TYPE.

Complex types, i.e. types in which at least one operator occurs, are called functors. Following the notational conventions of e.g. König (1990), a functor **a|b** is an ordered triple <**a**, |, **b**> where **a** is the value of the functor, | is the operation (i.e. one of \ and /) and **b** the argument. The head of a functor **a|b** is **a** if **a** is in BASE, and the head of **a** otherwise.

A weak calculus over TYPE is the so called AB system, named after Ajdukiewicz and Bar-Hillel. One way of defining this system is by stipulating three axioms and one rule of inference (capitals indicate sequences of types):

(3) (a) **x → x** (Identity)
(b) **y/x x → y** (Right Application)
(c) **x y\x → y** (Left Application)
(d) from **S → x** and **T x U → y**, infer **T S U → y** (Cut)

The proposition **a/b b c\a → c** is an example of a theorem of AB over TYPE. Its proof runs like this:

(4) **a/b b → a** (axiom of Right Application)
a c\a → c (axiom of Left Application)
a/b b c\a → c (cutting the first proposition into the second)

There is no way, however, to prove the slightly permuted proposition **b a/b c\a → c** in the AB calculus (3). Consequently, this proposition is not a theorem of AB. In fact, it is easy to see that the following generalization holds:

(5) If a proposition $\mathbf{a}_1 \ldots \mathbf{a}_n \rightarrow \mathbf{b}$ is a theorem of AB over TYPE, no permutation of distinct types $\mathbf{a}_i$ and $\mathbf{a}_j$ will yield a valid reduction to any type **c**.

This is an immediate consequence of the fact that under AB for any two types **x** and **y** there is at most one configuration in which they are reducible, and at most one type to which they can reduce; both restrictions can be read from the axioms

in (3). Since linear ordering and functor-argument relations are rigidly fixed in AB, the calculus is a very restrictive one. It computes a partial function from the set of n-tuples (sequences) over TYPE into TYPE.

To escape from the rigidity of the AB calculus, we can obtain more powerful systems by adding axioms and rules of inference. In particular, we may look for other rules in the spirit of the functional interpretation of complex categories. As a guideline we can use the apparent similarity between the two slash operators and the operation of arithmetical division, while considering concatenation as a non-commutative form of product of integers. Here are some candidates involving the left slash; right slash variations are easy to construct.

(6) (a) from $\mathbf{x\backslash y \rightarrow x\backslash y}$ and $\mathbf{z\backslash x \rightarrow z\backslash x}$, derive $\mathbf{x\backslash y\ z\backslash x \rightarrow z\backslash y}$
(b) from $\mathbf{x\ y \rightarrow z}$, derive $\mathbf{y \rightarrow z\backslash x}$
(c) from $\mathbf{x \rightarrow x}$ and $\mathbf{y \rightarrow y}$, derive $\mathbf{x \rightarrow y\backslash(y\backslash x)}$ and $\mathbf{y\backslash(y\backslash x) \rightarrow x}$
(d) from $\mathbf{x\backslash y \rightarrow x\backslash y}$ and $\mathbf{z \rightarrow z}$, derive $\mathbf{x\backslash y \rightarrow (z\backslash y)\backslash(z\backslash x)}$ and $\mathbf{x\backslash y \rightarrow (x\backslash z)\backslash(y\backslash z)}$.

These candidate rules all appear to express tautologies of arithmetics when we replace the arrow by the identity sign, abstract from directionality and interpret the type variables as integers. Pertaining the search for possible extensions of AB, we could also go for mixed rules, in which division is instantiated by both leftward and rightward directed slashes but the arithmetic connection, which of course abstracts from directionality, is preserved. We may, then, construct rules like

(7) (a) from $\mathbf{x/y \rightarrow x/y}$ and $\mathbf{z\backslash x \rightarrow z\backslash x}$, derive $\mathbf{x/y\ z\backslash x \rightarrow z/y}$
(b) from $\mathbf{(x/y)\backslash z \rightarrow (x/y)\backslash z}$ and $\mathbf{(x\backslash z)/y \rightarrow (x\backslash z)/y}$, derive $\mathbf{(x/y)\backslash z \rightarrow (x\backslash z)/y}$.

If a rule has only instances of the Axiom of Identity in its *from*-part, e.g. (6c), the *derive*-part of the rule qualifies as a possible axiom; otherwise, it can be added as a rule of inference.

A particular expression of the arithmetic "faithfulness" of categorial rules and calculi is formalizable by means of the Van Benthem Count of primitive types (see Van Benthem 1986):

(8) for all $\mathbf{x}$ in BASE, *x-count* is a function such that
if $\mathbf{y}$ in BASE, x-count($\mathbf{y}$) = 1 if $\mathbf{x = y}$
x-count($\mathbf{y}$) = 0 if $\mathbf{x \neq y}$
otherwise, x-count($\mathbf{y|\,z}$) = x-count($\mathbf{y}$) - x-count($\mathbf{z}$)
and x-count($\mathbf{y\ z}$) = x-count($\mathbf{y}$) + x-count($\mathbf{z}$).

Occurrences of primitive types in values of functors are recursively marked positive, occurrences in arguments of functors negative. So x-count($\mathbf{x/(y/z)}$) = x-count($\mathbf{x}$) = x-count($\mathbf{y/(z/x)}$) = 1, but y-count($\mathbf{x/(y/z)}$) = x-count($\mathbf{y/(x/z)}$) =

-1 and y-count(**x**) = 0. The common property of the rules in (6) and (7) is that every proposition occurring in the rule is such that for every primitive type **x** the x-count to the left of → equals the x-count to the right of the reduction operator. Clearly, if all the rules of a calculus have this property, all its theorems will have it to. Thus we may define *count invariance*:

(9) A categorial calculus is said to be count invariant iff for each theorem **S** → **y** and for each basic type **x**, x-count(**S**) = x-count(**y**).

Following this track, we can for a given set TYPE single out the set PCOUNT of propositions of the form (1) the left hand-side of which equals the right-hand side modulo x-count. If all the theorems of a calculus are within this class, the calculus is count invariant. It is easy to check that the AB calculus (3) has this property: the three axioms do, and the Cut rule (3d) is innocent in this respect, since its premises must be traced back to axioms and the derived proposition preserves the counts of the premises.

Van Benthem (1986) proves the count invariance of the L-calculus, presented in Lambek (1958), which essentially adds one axiom and one rule of inference to the AB system (3).

(10) (a) **(x y) z → x (y z)** and **x (y z) → (x y) z** (Associativity)
(b) from **x y → z**, infer **x → z/y** and **y → z\x** (Introduction)

Again we see that instantiations of the axiom will have equal sides modulo count. As for Introduction, the premise has to be traced back to the axioms and is therefore in PCOUNT by assumption; thus for all t, t-count(**z**) = t-count(**x**) + t-count(**y**), and consequently, t-count(**x**) = t-count(**z**) - t-count(**y**) and t-count(**y**) = t-count(**z**) - t-count(**y**), expressing the desired count properties of the derived propositions. Alternative set-ups for L, as e.g. in Zielonka (1981) and the system M of Moortgat (1988), do of course preserve count invariancy.

Although both L and AB are both within the realm of count invariance and the former seems a modest extension of the latter, L is considerably stronger than AB. The combined effort of Associativity and Introduction produces interesting theorems that are not available in AB, like:

(11) (a) **x → y/(y\x)** (Montague Lifting)
(b) **x/y y/z → x/z** (Geach Composition)
(c) **(x\z)/y → (x/y)\z** (Direction Switch; Associativity)

Proofs of these and other theorems of L can be found in Moortgat (1988), Van Benthem (1991) and other work by the same authors. The power of L is summarized by the property of structural completeness, proven for L in Buszkowski (1988b) and phrasable as:

(12) **(x y) z → t** if and only if **x (y z) → t**.

This amounts to the insight that derivability in L is not affected by the grouping of types and consequently any bracketing goes if some bracketing does. To push this line further: L computes a total function from the set of n-tuples over TYPE onto the powerset of TYPE. To see why, note that each n-tuple is a substring of infinitely many strings that are L-reducible to some type. Each n-tuple of types is, as a consequence of (12), therefore a `constituent' at the left hand side of infinitely many theorems of L and is as such reducible in the respective proofs of the theorems to some appropriate type, depending on its environment and the reduction hypothesis. This aspect of calculi of the L-kind is used in Steedman (1990) and Moortgat (1988) for the derivation of so-called non-constituent coordination, a topic I will discuss extensively in chapter 2.

L is neither omnipotent nor trivial. Infinitely many propositions over TYPE are not theorems of L and for each sequence of types there are infinitely many types to which it cannot be reduced. This restriction is induced by count invariance, which rules out e.g the propositions in (13), and by the direction sensitivity of the calculus, blocking the derivation of propositions like (14).

(13) (a) $\mathbf{x\ x \rightarrow x}$
(b) $\mathbf{x\ x/(x\backslash x)\ x \rightarrow x}$
(c) $\mathbf{x \rightarrow x\backslash x}$

(14) (a) $\mathbf{x/y\ z\backslash x \rightarrow z/y}$ (Mixed Composition)
(b) $\mathbf{x \rightarrow y/(y/x)}$.

At the linguistic side, König (1990: 38ff) mentions non-peripheral extraction, gapping and parasitic gaps as phenomena that pure L cannot cope with. The case of non-peripheral extraction is instructive. Moortgat (1988a) correctly remarks that we would like to derive the type of a fronted *wh*-constituent from the derivation of a sentence with the *wh*-constituent in its basic position. In general, we would like to be able to prove (15b) whenever we have a proof of (15a). To assure generality, we would need (15c), expressing the combinatorial potential of a fronted wh-constituent, which would guarantee the viability of (15b). (Capitals denote strings of types, whereas the asterisk is the concatenation operator that can be added to L).

(15) (a) $\mathbf{X\ wh\ Y \rightarrow s}$
(b) $\mathbf{wh\ X\ Y \rightarrow s}$
(c) $\mathbf{wh \rightarrow s/(X * Y)}$.

The problem here is not the presence of the concatenation operator, but the fact that **X** and **Y** are at different sides of the wh-constituent in (15a) — the theorem that should license the fronting — but at the same (right) side in the fronting case (15b). It is clear, however, by inspection of the calculus L, that none of its rules can affect the ordering of types (cf. Moortgat 1988). Consequently, there will not

be any proof in L leading us from (15a) to (15b): the general lifting (15c) is necessary but not available. We need, therefore, additional facilities.

If one feels severely limited by the properties of L, one could free the calculus from directionality by adding rule (16a) and from count invariance c.q. occurrence sensitivity by adding one or both of the monotonicity rules (16b) and (16c).

(16) (a) from **x y** → **z**, infer **y x** → **z** (Permutation)
(b) from **x** → **y**, infer **x x** → **y** (Expansion)
(c) from **x x** → **y**, infer **x** → **y** (Contraction).

Permutation gives rise to a direction insensitive calculus with theorems **x/y** → **x\y** , **x y** → **y x** and vice versa. For example, the relation between (15a) and (15b) is formatted by iteration of Permutation. Using L+Permutation (with the undirected division operator), Van Benthem (1991) outlines the logical perspectives of the various categorial calculi by reconstructing Application in Lambek style (17a) and Introduction (18a) as rules familiar from implicational logic, to wit Modus Ponens (17b) and Conditionalization (18b), respectively.

(17) (a) from **x** → **z**|**y**, infer **x y** → **z** (Lambek Application)
(b) from **p** and **if p then q**, infer **q** (Modus Ponens).

(18) (a) from **x y** → **z**, infer **x** → **z**|**y** (Lambek Introduction)
(b) from **p** and **q**, infer **if q then p** (Conditionalization).

L+Permutation, thus reconstructed as a propositional logic, is still weaker — in the sense defined above — than both the classical and the intuitionistic propositional calculi: L+Permutation is an occurrence logic, limiting Conditionalization to the withdrawal of exactly one particular assumption (cf. Van Benthem 1991: 40ff).

It is precisely for its intermediate position between rigid AB and full (ordering and occurrence insensitive) conditional logic that L qualifies as an interesting vehicle for linguistic description. Results concerning the recognizing power of categorial calculi endorse L's relevance as a complete, yet occurrence and ordering sensitive system. AB recognizes the context-free languages, as was proven in Bar-Hillel *et al.* (1960). Buszkowski (1988b) gives evidence for the conjecture that L has exactly the same power as AB if only the type raising consequences of L can be limited to finiteness per type. König (1990; part III) presents a construction to this effect for a given lexicon: its precompilation. In conclusion she postulates the required equivalence of L and context free grammar. But Van Benthem (1991: ch. 8) proves that stronger calculi tend to collapse into the regular languages; in particular, L+Permutation+Contraction is shown to recognize only these. Somewhere in between, the recognizing power might peak. The interesting conclusion, then, is that calculi beyond L are too permissive for natural languages, which are definitely not completely regular,

whereas L seems to add `only' strong capacity to the context-free AB, disguised as structural completeness (12). Since we may learn from today's generative grammarians that strong capacity is what counts, and Zwarts (1986), Moortgat (1988) and others call for new constituents along the classical `immediate phrases', L may serve us well in this respect - as it might also be a satisfying compromise in the backstage debate on the context-freeness of natural languages.

The issue of strong capacity opens the door for a subtle and maybe insightful classification of natural language phenomena. Given some assignment of types to lexical units, we could look for parameters in the derivation (or proofs) of terms and formulas in a language, more or less in analogy with the parametrization of sets of expressions with respect to the hierarchy of production grammars. In the latter setting, it is of considerable interest, for example, to prove that NP structure in a language as Dutch can be defined satisfactorily by a regular grammar, whereas VP structure requires at least an indexed context free grammar (cf. Pullum and Gazdar 1982). In the same vain, it is by no means trivial to see, as Van Benthem (1986; ch. 8) shows, that first order quantification can be handled within the realm of regular expressions. The L calculus constitutes a fine grained system of alternatives for a derivation that can be ranked by all types of criteria: length of a derivation, complexity of types involved, operators used, rules and axioms involved, and so on. Before we can take on this matter, some reflection on the assignment of types to lexical items is in order.

2. On the assignment of types

Studying the papers and books on categorial calculi and grammars, one hardly finds comments on the nature of the set of basic categories or types. In general, authors seem to agree that we need a basis of order two, but as a rule they do not argue for a restriction to this effect. Then, there is a long standing tradition, traced back to Leśniewski in Casadio (1988), of distinguishing the type of names (**n** or **e**) and the type of sentences (**s** or **t**), where one is free to consider the pair (**n**, **s**) of the Polish School as a syntax oriented way of labelling the basic types, and the pair (**e**, **t**) as the generator of a semantic algebra. The latter interpretation is enhanced by Montague's choice to leave the basic type **e** empty in his syntax: there are neither atomic nor complex expressions of this type. There may not even be any atomic expression of type **t**.

As a rare exception to the rule, Bach (1988) considers possible claims about Universal Grammar (" ... in Chomsky's sense ...") emerging from some fixation of the set of basic categories. For example, having **t** and **e** in the basic set would amount to the claim that (declarative) sentences and names are recoverable categories in every language. Quite intelligibly, Bach does not go deeply into this sort of considerations in which `atomic' and `fundamental' tend to be identical. For if we would search all languages of the world for common categories, we could find — depending on the standards we use — none or a whole lot of them. As an experiment: are you able to think of a natural language without quantifiers or a language without property expressions, and if you are not, are you sure that you can consistently isolate them as a (morpho-)syntactic category in each and every language? Nevertheless, there is some metaphysical attraction in having the pair (**t**, **e**) as the Ying and Yang of linguistics, where **t** stands for structural dependency and algebraic symphony, and **e** for autonomy, invulnerability and chaos. Clearly, one basic category is too little to satisfy our dialectic needs, but the question is whether two are enough.

If we do not attach `universal' claims to our choice of elementary categories, we are urged to provide a theory-internal motivation for the extension of the basis. One such motive may be looked for in the desirable systematic relation between syntactic categories and semantic types, within the realm of compositionality. As long as semantics is modelled in the logic of sets, we are well served with a restriction of basic types to entity and proposition, all other semantic objects being constructible from these. Since categorial grammar seeks this systematic relation by its very nature (cf. Casadio 1988), we have at least a syntactic program in keeping the mapping of categories on types as `isomorphic' as possible. But clearly, as soon as we leave the fields of atomic categories and atomic types, we switch from isomorphic relations to homomorphisms when attaching to syntactically (distributionally) distinct expressions as nouns and intransitive verb phrases one and the same type: that of sets or properties. Therefore, nothing is

lost if we disconnect syntactic and semantic type assignment altogether and attribute pairs of types to lexical items, one member indicating syntactic, the other semantic combinatorics, more or less in Montague style, along the lines suggested in Bach (1979) and Moortgat (1990). We may even have the syntactic and the semantic combinatorics computed in different calculi. To give just an example: we could decide to have the category **[np, e]** for proper names and the category **[np, t/(t\e)]** for quantificational NPs, while maintaining something like **[s\np, t\e]** for VPs and **[(s\np)/np, (t\e)/e]** for finite transitive verbs; additionally, we may assume that the syntax is regimented by the calculus L, whereas the semantics should not be sensitive to ordering, but just to functional relations. Incorporating this format into the rules of grammar, then, we would have the following schema:

(19) **[x, a] [y, b]** → **[z, c]** iff **x y** → **z** in L and **a b** → **c** in L+Permutation.

This schema is instantiated, e.g., by

(20) (a) **[np, e] [s\np, t\e]** → **[s, t]**
(b) **[np, t/(t\e)] [s\np, t\e]** → **[s, t]**
(c) **[(s\np)/np, (t\e)/e] [np, e]** → **[s\np, t\e]**
(d) **[(s\np)/np, (t\e)/e] [np, t/(t\e)]** → **[s\np, t\e]**.

Of course, we may also choose other calculi for the syntactic or the semantic combinatorics. Compositionality is warranted, and flexibility is our gain. Doing so, there is also room for assigning a primitive type to predicates, for example, which is argued for semantically in Chierchia and Turner (1988).

Some authors, notably Lambek (1988) and Buszkowski (1988a), do not make any claims on the extension of the set of primitive types, although Lambek explicitly mentions **s** and **n** among them. They keep open the possibility of a larger set of basic categories. Indeed, when we interpret categorial grammar as a free algebra (Buszkowski, Moortgat, Lambek), little or nothing hinges on the extension of the set of basic types, as it seems, although the relevant count is of the one/two/many species. Besides, Ponse (1987) proved that every language that is categorially recognizable, can be recognized by using only one primitive type. There is a trade-off between the number of primitives used, the complexity of types occurring in the lexicon and the derivations, and the degree of linguistic transparency. Hence, things become different when categorial calculi are exploited for linguistic and/or computational purposes. In that case, the order of a grammar may become crucial, as König (1990) shows, and may provide a second motive for enlarging the categorial basis. Order of types is defined throughout the literature in the following way.

(21) order(**x**) = 0, if **x** is basic
order(**x/y**) = order(**x\y**) = maximum(order(**x**), order(**y**) + 1)

The order of a grammar, then, is the highest order of any type assigned in the lexicon, which holds the initial assignment of types to expressions. This means that the order of a grammar is defined by the way basic categories are used in forming classes of expressions. A grammar of order zero has exactly as many `word classes' as there are basic categories. No word belongs to any functional category a priori. The simplest unidirectional grammar of higher order, order one, makes available for the initial assignment an infinite number of types of the form $\mathbf{(...(x_1|x_2)|...|x_n)}$ with $\mathbf{x_i}$ in BASE, e.g. **s/n**, **n\s**, **(s/s)/s**, **(s\s)/n**, **((n/s)/n)\s**, **((s/n)\s)/n**, etc. According to (21) these categories have in common that on every level of recursion an argument is a type of order zero. Upgrading the grammar to order two, we can add all types in which arguments contain exactly one division. This move will not augment lexical discernment, since the lexicon is finite by definition.

The combination of low order of a grammar and a minimal set of basic categories is a serious threat to linguistic relevance. To see why, let us take a closer look at a bidirectional grammar of order one on a binary basis, e.g. the set {**s**, **n**}. Constructing, in a thus constrained grammar, a proper categorial distinction into names, properties and determiners — three fundamental types of expression in sentential strings — turns out to be inadequate. The argument runs as follows. One basic category is assigned to sentences; none of the three types of expressions is a sentence, so only one of them may be assigned an atomic category. One of the other two could be an elementary function **x|x**. Note that directionality of operators, though available for distinguishing categories, is not the right instrument for making fundamental differences (cf. Flynn 1983 where directionality of categories is construed as a language specific parameter). The third category takes either of the two others to sentences, since these can be formed by a single property, a single name and a determiner. This intended category must be the atomic one, or else the resulting function **s|(x|x)** exceeds the bounds of order. The third category therefore is of type **s|n**. This leaves the category **n|n** for the second, expressing that one of name, property and determiner is an automorphism. Consequently, for this type assignment to be linguistically meaningful, one of the three following statements must hold:

(22) (a) a combination of a name and a determiner is a name or a determiner
(b) a combination of a name and a property is a name or a property
(c) a combination of a determiner and a property is a determiner or a property.

Of these, only the first makes sense as a generalization. It implies that a nominal determiner phrase is categorially identical to one of its (possible) parts. Since categories are characterized by intersubstitutability (Buszkowski 1988a), the nominal determiner phrase should be endocentric. As far as this phrase may be said to be endocentric, it is so by taking the determiner as the head. In that case, the determiner belongs to an atomic category and the name or noun is the **n|n**

functor. This implies that a determiner of type **n** and a property of type **n**|**s** can be combined to form a sentence : in fact, determiners turn out to be syntactically what we naively thought names were. There is little chance, however, for satisfying combinatorics.

The above argument yields a minor but instructive generalization;

(23) Given a first order grammar on a basis of order two and three expressions A, B and C such that ABC is a sentence, one of the atomic categories is endocentric.

Clearly, if one of A, B or C is a sentence itself, propositions form the endocentric category. But propositions are not endocentric, by definition. Hence, as soon as we find a language whose propositions may be built from three different types of expressions and that has more than one exocentric category, a grammar of order one on a basis of order two induces a somehow unsatisfactory lexicon for that language. I call a lexicon unsatisfactory, in relation to the grammar operating on it, if it is combinatorially opaque in the following sense:

(24) An initial type assignment I to expressions $A_1,...,A_n$ is **combinatorially opaque** with respect to a calculus C if there is a proof of $\mathbf{a}_1 \ldots \mathbf{a}_i \ldots \mathbf{a}_n \rightarrow \mathbf{t}$ in C with $\mathbf{a}_j \in I(A_j)$, and every proof of that proposition is also a proof of either
some permutation $\pi(\mathbf{a}_1 \ldots \mathbf{a}_i \ldots \mathbf{a}_n) \rightarrow \mathbf{t}$, or
some proposition $\mathbf{X} \rightarrow \mathbf{t}$ where degree(**X**) > degree($\mathbf{a}_1 \ldots \mathbf{a}_n$).

For this purpose, let degree be defined by the recursion

(25) degree(**x**) = 1 if **x** is basic
degree(**x**|**y**) = degree(**x y**) = degree(**x**) + degree(**y**).

Under this proviso, the impact of (24) is that for a lexicon to be transparent there must be at least one derivation of a provable sequent that does not resort to permutation, type raising or addition of types — the latter two are degree increasing. This means that a lexicon is unsatisfactory, in relation to a grammar of certain order, if some fundamental functional relations between expressions cannot be read from the order-restricted assignments in the lexicon alone. For example, the functional discontinuity provoked by *wh*-binding should not be restored in the grammar exclusively by lifting *wh*-elements; instead, the elements in a transparent lexicon are to be typed such that there is one derivation of wh-sentences not resorting to type lifting theorems like (11a). This strategy would rule out e.g. the treatment of *wh*-binding of Moortgat (1988a), pointed at in (15). Under that strategy, the only proof of (26) involves lifting of the *wh*-element and (restricted) permutation. Transparency of the initial assignment in the sense of (24) can only be claimed if the grammar also allows a proof of (26) without lifting or permutation, for *wh*-elements hold the front position of sentences canonically.

(26) **wh T → s**

Transparency would not be violated, however, by an application of Moortgat's lifting device to cases of stylistic fronting: a grammar that handles wh-constructions transparently, may be assumed to handle stylistic fronting in a similar way, and the mere existence of an alternative derivation with lifting is harmless in that case. (24) thus implies that — under a combinatorially transparent lexicon — rules that raise the order of a category, only serve to give alternative analyses. One could even say that whenever a Lambek grammar is said to be `invariant under arbitrary arrangement of functor-markers' (Buszkowski 1988b: 81), some initial assignment of function markers is assumed: one given by a transparent lexicon, expressing canonical functor/argument structure. If, on the other hand, a lexicon is combinatorially opaque as defined in (24), it may hardly steer a data driven grammar. The moral of transparency is this: the relation between a lexical assignment and a grammar fed by it must be such that for every grammatical string at least one derivation monotonically decreases the degree of the string.

Grammars of order one have certain features that are useful in linguistic computation, as König (1990; p. 101) notes in her discussion of the problem of spurious ambiguity for L-parsing. The reasoning in this section shows that grammars of order one for natural language are to be based upon an extended set of primitive categories, giving rise to “flat many-sorted models” (Van Benthem 1991: 95).

3. On minimal categorial grammar

3.1. Combinatory principles of minimality

It is convenient to describe a grammar G of a language T as a triple of an assignment A of categories to a lexicon, a categorial calculus CC which essentially consists of rules of inference and/or axioms, and a set of designated categories D, possibly a singleton. Such a grammar G(T) = <A, CC, D> meets the all-and-only requirement: it defines a bipartition of the set of strings over the lexicon of T, one subset being equal to T, the other being equal to T's relative complement. Most of the work referred to until now, can be said to bear on the all-part of the requirement. It provides the means for assuring that for every possibly wellformed string there is a derivation. To deal with the only-part of G's task, we have to select rules of type change or to restrict some of them. The modal operators of Hepple (1990) and Morrill (1990), for example, are added to the lexical assignment in order to impose a certain derivational regime, thus restricting the set of possible outcomes. In particular, these operators may serve to define barriers for discontinuity. Moortgat (1988) introduces specialized operators to handle extraction and infixation. We will need this type of enriched categories, restricting derivations but enlarging expressive power, to ensure adequacy. Another, though not necessarily divergent way of bounding the capacity of a calculus, is making some choice from the combinatorial possibilities a calculus offers. The latter strategy is presented by Ades and Steedman (1982) and subsequent work by the second author as Combinatory Categorial Grammar, an application of the logic of Curry and Feys (1958). In such a system, the grammar is not made up of more or less abstract rules of type change, as in the Lambek calculus, but by explicit combinatorial operations. To give an example: Steedman (1990) suggests that the categorial grammar of Dutch contains the following rule, akin to Mixed Composition (14):

(27) **x/y y\...z → x\...z** (Dutch Forward Crossing Composition)
where **y\...z** is a generalized verb.

The rule is meant to take care of Dutch verb raising. It would account for the grouping of the typed phrases in the following construction.

(28) Jan heeft geen van hen het onheil kunnen melden.
vp/vp **vp\np\np**
Jan has none of them the calamity been-able announce
`Jan has not been able to announce the calamity to any of them.'

Verb raising will be discussed more extensively in section 4. Let us concentrate here on the shape of a categorial grammar that contains rule (27). That grammar would contain also the basic rules of Application (3b) and (3c); these rules are likely to take care of the remaining combinatorics in (28). None of them,

however, is taken to be a consequence of an underlying system of type change. They are chosen as principles, rather than being theorems.

This combinatory approach offers an intriguing perspective: classifying languages in terms of the categorial combinatorics needed to grammaticize that language with respect to a certain lexical assignment of categories. For example, suppose that Dutch, but not German, needs a Forward Crossing Compositional Rule of type (27). We could say, then, that Dutch is a crossing compositional language and German is not, meaning thereby that under every transparent assignment of categories to the Dutch lexicon at least some sentences of Dutch can only be derived through some form of Crossing Composition. Yet, this is not very insightful. To exploit this perspective and to work towards some alternative, categorial classification of languages, we need two more notions: (i) the reductional degree of a rule (30a) and of a set of rules (30b), and (ii) the notion of minimal grammar (31). In order to facilitate the formulation of these notions it is assumed that combinatory rules are axioms of the form $\mathbf{X} \rightarrow \mathbf{Y}$, that the degree of (sequences) of types is as defined in (25), and that the only rule of inference in a grammar is the Cut rule, expressing the transitivity of type change:

(29) from $\mathbf{X} \rightarrow \mathbf{Y}$ and $\mathbf{Y} \rightarrow \mathbf{Z}$, infer $\mathbf{X} \rightarrow \mathbf{Z}$.

Since this rule is a harmless inference in that it does not add anything to the nature of a grammar, it will be neglected in the following exposition.

(30) (a) The **reductional degree** of a combinatory rule $\mathbf{X} \rightarrow \mathbf{Y}$ is the difference degree(**Y**) - degree(**X**).
(b) The **reductional degree of a set of rules** is the average of the reductional degrees of its members.

Obviously, the reductional degree of reducing rules like Application and Composition is negative while the degree of type lifting rules is positive. Consequently, the reductional degree of a set of rules containing the latter will be higher than the reductional degree of a set that is free from type lifting rules.

Given these definitions and the proviso that the categorial calculus of a grammar is formatted as a set of axioms driven by the Cut rule, the spirit of minimality may be clear: the more reductional a categorial calculus is, the greater the chance that the grammar it constitutes is minimal for the language L. But minimality should also extend to the number of rules stated by the grammar: intuitively, a grammar with one rule of reductional degree -1 is more qualified to be minimal than a grammar with ten rules of that reductional degree, even though the corresponding sets are, of course, of the same reductional degree. This consideration is also captured in the following definition.

(31) A grammar G = <A, CC, D> is **minimal** for a language T with respect to A iff CC is the smallest set such that for every grammar G′

= <A, CC′, D> of T the reductional degree of the set CC′ is greater than or equals of the reductional degree of CC.

(31) introduces minimality of a grammar for a language with respect to a given lexical assignment as a function of reductional degree and cardinality. In the definition, the lexical assignment is kept constant. One may wonder whether this is an adequate assumption. I argue that it is. As a typical case, consider the question of the categorization of Dutch finite verbs. Their subject argument can occur to the right as well as to the left. Moreover, the relative position of other nominal arguments may vary. Here is a sample of types for the transitive finite verb *koopt* `buys' with sentences they match; the subject argument in the type and the subject in the sentences are in italics.

(32) (a) **s/np/*np***
Koopt *Jan* deze dichtbundel?
buys Jan this book-of-poetry
Misschien koopt Jan deze dichtbundel
maybe buys Jan this book-of-poetry

(b) **s*np*/np**
Jan koopt deze dichtbundel.
Jan buys this book-of-poetry

(c) **s\np*np***
...dat *Jan* deze dichtbundel koopt
that Jan this book-of-poetry buys

Although the variation is predictable rather than stylistic, there is no way of deriving the three types from each other in a directed categorial framework. A rule **s/.../np ↔ s/...\np...** in the grammar of Dutch would require full permutational force, which is certainly more than Dutch can stand. So we have no choice: all three types must be assigned lexically, even against the background of L (cf. König 1990: 39). This assignment, then, expresses facts of Dutch that are beyond the combinatorial power of an adequate categorial grammar, minimal or non-minimal. Yet, there may be different opinions on these facts or their proper representation. Even if we are faced with essentially different assignments, we may compare the grammars fed by them: if all are minimal with respect to their assignment and we cannot decide which assignment is the most appropriate, we may select as a minimal grammar the one with the most reductional rule set.

It follows from the nature of types and type assignment that every adequate grammar, be it a minimal or a non-minimal one, for a non-trivial language will contain the axioms of functional application. In fact, we may assume that the final step in every derivation of a wellformed formula of a designated type involves Application if designated types are basic - a non-controversial prerequisite. Therefore, every categorial grammar is **applicative**. Hence, every minimal adequate grammar for language T is applicative, and we may also call applicative

every language described by some categorial grammar. As for other possible axioms, the choice is restricted, not by any established calculus, but by one fundamental principle, elaborated on e.g. in Steedman (1990, 22). I will state it here as a requirement of the format of axioms.

(33) DIRECTIONALITY
For any axiom of the form **X** → **Y**, if **a** is an argument in some type of **X** and some type of **Y**, **a** is slashed equally in **X** and **Y**.
(**a** is slashed equally in types **b/a** and **c/a**, and also in types **b\a** and **c\a**)

The principle expresses that directionality is a characteristic and inalienable property of occurrences of arguments. It underlies what Steedman calls "directional consistency" and "directional inheritance" of combinatory rules. It is not to be confused with the parametrized directionality of functors, proposed by Flynn (1983). (33) is just an explicitation of a weak form of preservation of order, and it prohibits permutation from being introduced by accident. To the same effect, Steedman (1990) notes that any combinatorics is to be limited by some form of adjacency: the inference engine of the grammar is supposed to instantiate the types in the antecedent of the axiom schemes only by adjacent types in the sentence, in the order given by the scheme. This restriction, too, prevents permutation from slipping in. Additionally, adjacency can be transformed to a numerical condition on the antecedent of axioms: if operations are to effect only adjacent categories, it is no good to have axioms with antecedent strings of more than two members. To see why this is so, consider a hypothetical axiom with an antecedent of length three:

(34) **a b c → D**

Two cases are relevant: one of **a**, **b** or **c** is also in the string **D**, or none is. In the latter case it is clear that either the rule somehow affects **a** and **c**, thus violating adjacency, or the rule can be split in two independent operations. Consider e.g. the hypothetical axiom

(35) **x/y z\x/w w → z/y**

Here it is obvious that the rule just compiles one rightward Application and one leftward Mixed Composition. Next, consider

(36) **x/y z\x/w w\v → z\v/y**.

There are two options to justify this rule. First, it is derived from a general type of permutation of the following kind:

(37) **z\x\y → z\y\x**.

This would license the effect of (36) pairwise: mixed composition to the right, yielding **z\x\v**, retyping the result as **z\v\x** according to (37) and then mixed

composition to the left. The second justification would amount to a particularization of the effect of (36) to the configuration of types stipulated in the antecedent. In that case, however, the relation between two adjacent categories — for example **x/y** and **z\x/w** — is determined by a third category that is not adjacent to both of the others. Since adjacency is not transitive, this would imply that the **a** and **c** of (34) influence each others domain, and this can be seen as a violation of the adjacency requirement.
In the other case relevant for the evaluation of rule type (34) one of the antecedent types is mere context to some operation on the other two. If **b** is the untouchable type, adjacency is again violated again. If **a** or **c** survives the operation, **D** is, e.g., of the form **a D′**. This string has to be reduced somehow. If **D′** is in any way traceable to **b c**, and **a** is supposed to hold a necessary condition for the operation, **D′** and **a** must have combinatory characteristics that could also have been encoded as type restrictions on **b** or **c** or both. But then **a** can be dispensed with in the formulation of the rule. In general, then, the spirit of adjacency appears to be directed against rules of the (34) species.

3.2. Minimal systems

3.2.1. Comparing rules

Directionality, count invariancy and adjacency yield a format for a notion of `possible rule of a nonpermutational, occurrence sensitive combinatory categorial grammar':

(38) A → B is a possible rule of grammar (axiom) iff
- (a) for all types **x** x-count(A) = x-count(B)
- (b) $|B| \leq |A| \leq 2$
- (c) the rule observes Directionality (33).

Now we can go for other combinatory rules in a minimal grammar. Among the candidates are all types of Montague Lifting, all kinds of Composition and all kinds of Division:

(39)			**x → y|(y|x)**	(Generalized Montague Lifting)
	(a)	(i)	**x → y\(y/x)**	(Harmonious Lifting)
		(ii)	**x → y/(y\x)**	
	(b)	(i)	**x → y/(y/x)**	(Disharmonious Lifting)
		(ii)	**x → y\(y\x)**	
(40)	(a)		**x|y y|z → x|z**	(Generalized LR Composition)
		(i)	**x/y y/z → x/z**	(Harmonious LR Composition)
		(ii)	**x/y y\z → x\z**	(Disharmonious LR Composition)
	(b)		**y|z x|y → x|z**	(Generalized RL Composition)

		(i)	$\mathbf{y\backslash z\ x\backslash y \rightarrow x\backslash z}$	(Harmonious RL Composition)
		(ii)	$\mathbf{y/z\ x\backslash y \rightarrow x/z}$	(Disharmonious RL Composition)
(41)	(a)		$\mathbf{x \mid y \rightarrow (x \mid z) \mid (y \mid z)}$	(Generalized LR Division)
		(i)	$\mathbf{x/y \rightarrow (x/z)/(y/z)}$	(Harmonious LR Division)
		(ii)	$\mathbf{x/y \rightarrow (x\backslash z)/(y\backslash z)}$	(Disharmonious LR Division)
	(b)		$\mathbf{x \mid y \rightarrow (z \mid y) \mid (z \mid x)}$	(Generalized RL Division)
		(i)	$\mathbf{x\backslash y \rightarrow (z\backslash y)/(z\backslash x)}$	(Harmonious RL Division)
		(ii)	$\mathbf{x/y \rightarrow (z/y)/(z\backslash x)}$	(Disharmonious RL Division)

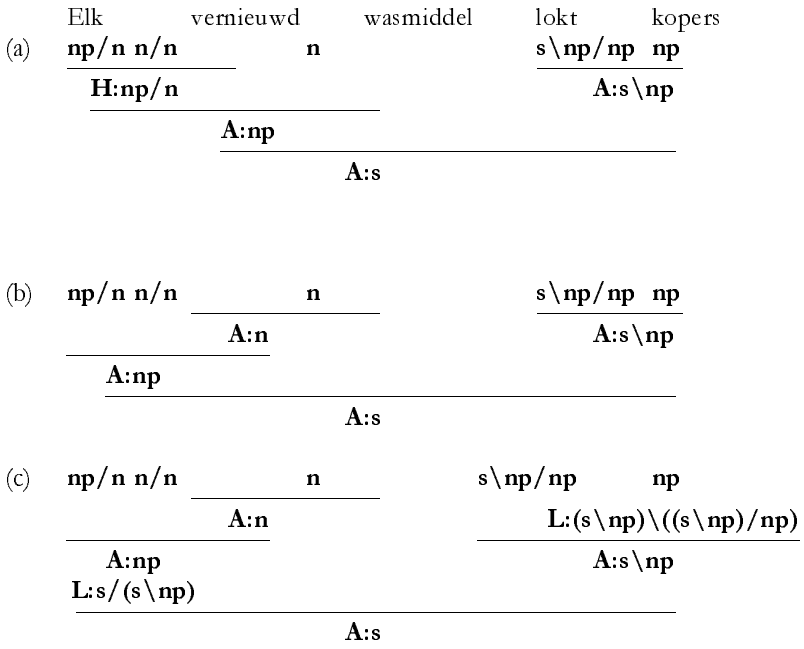
All these rules meet the conditions of (38). Yet, they are diveded into two classes with respect to minimality. On the one hand, we have the lifting rules and the rules dubbed `harmonious'. They provide alternative analyses for a directed first order application grammar; they are theorems of Lambek's slash introducing and slash eliminating calculus, which is structurally complete, as noted before. On the other hand, we find the `disharmonious' rules, which are in the complement of a Lambek calculus. Each of them assigns structure to strings of types that could not be reduced in a purely applicative way.

The crucial conjecture here is the following:

(42) Given a first-order assignment of types in the lexicon, $\mathbf{x_1 \ldots x_n \rightarrow a}$ (for some designated primitive **a** and with $n > 1$) is derivable in a Lambek driven grammar only if there is at least one pair $(\mathbf{x_i}, \mathbf{x_{i+1}})$ such that either $\mathbf{x_i = u/v}$ and $\mathbf{x_{i+1} = v}$ or $\mathbf{x_i = v}$ and $\mathbf{x_{i+1} = u\backslash v}$.

Note that we may assume without loss of generalization that the assignment of types to lexical items is within the borders of first order if we also assume that the assignment is finite and variable-free. The conjecture is easily proved by induction on the length of the antecedent string. Suppose n = 2. Obviously, $\mathbf{x_1\ x_2 \rightarrow a}$ iff one of $(\mathbf{x_1}, \mathbf{x_2})$ is a functor with head **a** and the other is the appropriate argument for that functor. In fact, count-invariancy requires for any antecedent-string $\mathbf{x_1 \ldots x_n}$ in a hypothesis $\mathbf{x_1 \ldots x_n \rightarrow a}$ there to be at least one $\mathbf{x_i}$ with head (i.e. positive occurrence of) **a** of the form $\mathbf{a \mid \ldots \mid y}$. If its neighbour on the right side is an appropriate argument, $\mathbf{x_i}$ and that neighbour are the pair in the iff-part of (42). If the neighbour to $\mathbf{a \mid \ldots \mid y}$ is not **y**, somewhere in the wanted direction there must be a functor with head **y**, again because of count-invariancy. Since the assignment is first order, **y** is a primitive type, and therefore somewhere to the right or left of $\mathbf{a \mid \ldots \mid y}$ a subsequence $\mathbf{x_j \ldots x_{j+k} \rightarrow y}$ must be derivable. Now the same reasoning as before holds for this smaller sequence. Either we arrive at some subsequence with a two-typed antecedent, giving the pair that is claimed in (42), or there is no derivation at all, since some necessary condition $\mathbf{Y \rightarrow y}$ (**y** basic) cannot be fulfilled. This proves the conjecture (42). The issue is illustrated in the following tree-style derivations; the markers at types indicate their derivational origin: A(pplication), L(ifting) or H(armonious Composition).

(43) *each* *innovated* *detergent* *tempts* *buyers*



The derivation (43b) is the minimal one: only Application is used to derive **s**. The other two show involvement of Harmonious Composition and Lifting, respectively. Neither is necessary to derive **s** which itself is always reached in a final applicational step.

In addition to the general fact that in every Lambek derivation of $\mathbf{X} \rightarrow \mathbf{a}$ with **a** primitive application is involved, the reasoning proving (42) implies that under first order assignment conditions the derivability of $\mathbf{X} \rightarrow \mathbf{a}$ depends crucially on the existence of a derivation in which *only* application is involved, `starting' with the applicative reduction of the pair claimed in the only-if part of (42). Hence, we have a minimality result for the class of languages — probably the context-free languages — that can be recognized by a Lambek grammar, given the assumption that lexical assignment is finite, variable-free and thus first-order constructible:

(44) The minimal grammar for the class of Lambek recognizable languages is the purely applicative AB.

Therefore, a Lambek grammar is minimal for no language at all. So we are entitled to skip all the harmonious axioms in (39) through (41) as candidates for rules of minimal grammar.

The result in (44) is not very surprising in the light of the conjectures mentioned above that AB and L are weakly equivalent. Van Benthem (1991: ch. 8) observes

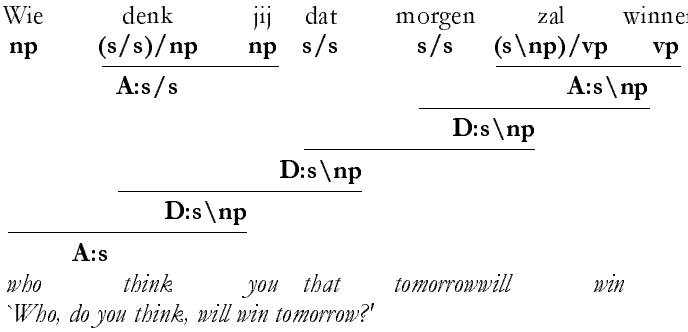
that the recognizing power of categorial systems in a hierarchy reaching from AB at one end to Lambek+Permutation+Contraction+Elimination at the other tends to fall down from context-free to regular languages, but does not exclude the possibility that the recognizing power peaks somewhere in between. If so, we can be certain that the peak is not won by adding structural rules to L. This type of rules (Permutation, Contraction, Elimination) ignores precisely those elements of linguistic fine structure, viz. complex positional and occurrence dependencies, that characterize languages outside the context-free realm. Added to the strong feeling that L itself does not exceed the weak power of AB, this may have provoked Van Benthem (1991: 95) to challenge "the Establishment in Mathematical Linguistics": maybe the Chomsky Hierarchy is not the measure of all things and the Categorial Hierarchy defines a new paradigm. Yet, we may be certain that to the extent (natural) languages appear to incorporate phenomena beyond the reach of context-free grammar, they have to be covered by categorial grammars from outer space. This is the message of (44): if we find language structures that we are not able to describe with finite means by providing an applicative grammar with functional types, no Lambekian extension of that grammar will help us out. As a matter of fact, then, the recognizing power of a categorial system can only be upgraded under the conditions of (38) by adding disharmonious rules.

Among the disharmonious rules, the disharmonious composition rules (40aii) and (40bii) are the most likely to figure in a minimal grammar for non-context-free languages, since their reductional degree — as defined in (30a) — is negative. Moreover, composition and division rules do the same job, as Moortgat (1988) shows, so that there is no need for including both in a minimal grammar. Disharmonious composition allows an argument to occur non-adjacent to the functors introducing them at the, for that argument, canonical side of the functor; disharmonious composition obeys directionality. The slight permutation introduced by this rule relative to a purely applicative ordering, is not only restricted by directionality, but also by the nature of the types intervening between functor and argument. (Because the Lambek calculus allows for flexible functor/argument patterns and has no fixed types, addition of disharmonious composition to this calculus yields a totally permutational calculus; cf. Moortgat 1988). The intervening types must form a head-argument chain that functions as a bridge for the transport of the dislocated argument. Therefore the only two language configurations for which disharmonious composition adds to the power of the AB grammar are strings of types that can be analyzed as (45a) or (45b):

(45) (a) **... a x|./y1 y1|./y2 ... yn|./b b\a ...**
(b) **... a/b yn|.\a ... y1|.\y2 x|.\y1 b ...**

A possible application of these bridges can be found in *wh*-movement from argument positions:

(46)



Here we see a chain of rightward directed functions that is able to transfer the missing subject argument of the embedded sentence to the left periphery of the matrix. The configurations in (45) may have been stated inaccurately, however. If we inspect the operation of disharmonious composition more closely, there appears to be a flaw in its formulation. In (45) the bridging functors are supposed to meet only one condition: their peripheral argument must be the head of some adjacent functor. In so far as they are to instantiate the main functor in a rule of disharmonious composition, this one condition suffices. But by instantiating the main functor, they may also have to instantiate the subordinate functor in a next application of disharmonious composition. In the latter role, it is unclear what is going to happen to their `own' arguments, i.e., those arguments that are not involved in the transfer or dislocation of "alien" arguments. In order to discuss the problem, we need a more precise notion of type under first-order assumptions. At the onset of this chapter, a type was introduced as a triple of head, operator and argument. Since disharmonious composition is directional, however, we must be able to distinguish between left and right arguments. To do so, let us define a first-order type as a structure of the following format:

(47) <primitive head, sequence of LEFT arguments, sequence of RIGHT arguments>.

The type **a/b**, then, stands for **<a,[],[b]>** and the more complex **a/b\c/d** abbreviates **<a,[c],[d b]>**. The sequences of arguments are ordered from peripheral to innermost argument at one side, i.e. we may rephrase Application in the following way:

(48) (a) **<x, L, [y Rrest]> <y, [], []> → <x, L, Rrest>**
(b) **<y, [], []> <x, [y Lrest], R> → <x, Lrest, R>.**

As for (disharmonious) composition, we have to deal with, e.g., a main functor of type **<x, L, [y Rrest]>**, as in (48a), and a subordinate functor **<y, L′, R′>**. In case of harmonious composition, the destination of **L′** is a problem, as is the question whether or not **R′** is inherited by the composed function. For disharmonious composition, we have similar questions, but with the strings

reversed. Let us review some possibilities for harmonious composition with the main functor occurring to the left.

(49) (a) **<x, L, [y Rrest]> <y, L′, R′> → <x, [L L′], [Rrest R′]>**
(b) **<x, L, [y Rrest]> <y, L′, R′> → <x, [L′ L], [R′ Rrest]>**

R′ is here the string of arguments delivered to the composed function by the subordinate functor. Since harmonious composition does not involve permutation of any kind, **R′** must become peripheral and hence `first-to-meet' in the new function; therefore, (49b) is the more adequate alternative with respect to the new right arguments string. It implies that the composed function cannot cancel any of the right arguments originating from the main functor, **Rrest**, until all the righthand arguments originating from the subordinate functor, **R′**, have been dealt with. As for the left arguments, no such reasoning is valid. In either case (49a) and (49b), some string of arguments will have to occur dislocated, i.e. non-adjacent, with respect to the function that introduces them negatively. Since only **L** — the main functor's left argument string — has a possibility to occur adjacent to its `seeking' functor, we must conclude that **L′** has to be empty in order to preserve the anti-permutational ideology of harmonious composition. For only if **L′** is empty, Harmonious Composition will not imply dislocation of any kind. So, in the format (47) the rule of harmonious composition for rightwards directed main functors is stated as in (50a); by the same reasoning, its leftward oriented counterpart must be like (50b):

(50) (a) **<x, L, [y Rrest]> <y, [], R′> → <x, L, [R′ Rrest]>**
(b) **<y, L′, []> <x, [y Lrest], R> → <x, [L′ Lrest], R>**

Note that this restatement of (40ai) and (40bi) contains no new restrictions. It just makes explicit the idiosyncrasies of harmony in a first order setting.

3.2.2. Generalizing the rules

The argument that lead us to the formulas in (50), also sheds light on the nature of the disharmonious composition rules. Disharmonious Composition is a restricted permutation rule under directional conditions, but should not have any side-effects on other parts of the string. The most restricted and elementary form of permutation is the position switch of two adjacent atoms in a sequence. Consider the configuration (51a). It is the canonical ordering of the atoms with respect to their types. The slightest possible permutation that respects directionality, is the switch of **b\a/c** and **d**, yielding the sequence (51b). Any other dislocation of **d**, as in (51c), is less elementary: if we assign position numbers to the atoms in each sequence, we see that in (51b) only two atoms have different numbers relative to the canonical ordering (51a), whereas in (51c) three atoms appear to have been moved.

(51) (a) **a b\a/c d c\d/f f**
(b) **a d b\a/c c\d/f f**
(c) **d a b\a/c c\d/f f**

If we want to consider Disharmonious Composition as the generator of the most restricted form of permutation, it is to license only (51b), and not (51c). But then it is also the natural representation of crossing dependencies: in (51b) the paths between the functors and their left arguments cross, whereas in (51c) they are nested. So, under the assumption that Disharmony introduces only elementary permutation with respect to the canonical adjacency of a functor and its arguments, it is appropriate to define Disharmonious Composition in such a way that the arguments of the subordinated functor are to be met first:

(52) (a) **<x, L, [y Rrest]> <y, L′, R′> → <x, [L′ L], [R′ Rrest]>**
(b) **<y, L′, R′> <x, [y Lrest], R> → <x, [L′ Lrest], [R′ R]>**.

Both of the argument lists of the subordinated functor **<y, L, R>** are peripheral in the argument lists of the composed function. Stated this way, Disharmonious Composition induces crossing dependencies. It thereby adds to the recognizing power of a categorial system if the rule can be adopted without turning it into a full permutational system.

At closer inspection of the formulas (52) we may conclude that disharmonious composition happens to be a form of bidirectional composition, harmonious to one side, disharmonious to the other. Interestingly, the three rules of Application, Harmonious Composition and Disharmonious Composition can be reformulated as decreasingly restrictive instances of one general rule format:

(53) Generalized Directional Categorial Rule (GDCR)
(a) **<x, L, [y R]> <y, L′, R′> → <x, [L′ L], [R′ R]>**
(b) **<y, L′, R′> <x, [y L], R> → <x, [L′ L], [R′ R]>**
Applicative: both **L′** and **R′** are empty
Harmonious: **L′** is empty (case 53a), **R′** is empty (case 53b)
Disharmonious: otherwise.

Note that the operation $[\mathbf{X}_1 \ldots \mathbf{X}_n]$ on argument lists is meant to be simple list concatenation defined as:

(54) conc(**[]**,**L**) = conc(**L**,**[]**) = **L**
conc($[\mathbf{x}_1 \ldots \mathbf{x}_n]$, $[\mathbf{y}_1 \ldots \mathbf{y}_m]$) = $[\mathbf{x}_1 \ldots \mathbf{x}_n\ \mathbf{y}_1 \ldots \mathbf{y}_m]$
conc($\mathbf{L}_1$,...,$\mathbf{L}_m$) = conc($\mathbf{L}_1$, conc($\mathbf{L}_2$,...(...conc($\mathbf{L}_{m-1}$, $\mathbf{L}_m$)...)...))

Thus, the GDCR can be thought of as the cancelling of a peripheral argument (the primitive head of the subordinated functor) under adjacency conditions and the concatenation of the remnant argument strings according to the recipe conc(X-arguments subordinate functor, X-arguments main functor).

Looking at the GDCR this way, even more general variants come into mind. For example, one could also parametrize the relative position of the argument-to-cancel in the main functor's argument string, or the concatenation mode, each move giving rise to new but restricted permutational variety. Other strategies may relativize the conditions with respect to the nature of one or both of the functors' heads or with respect to the main functor's argument lists. Exercises like these may serve to explore the space between AB and a system with full permutation, like the Lambek calculus enriched with Disharmonious Composition. Whereas any language will be overcharged by a fully permutational grammar, almost every natural language embodies some ordering variation. A well founded programm of minimizing categorial grammar is likely to be a good starting point for charting the extensions and intensions of such a variety.

When does it make sense to call a language disharmonious? The diagnostic configurations for disharmonious permutational languages are reminiscent of those in (45), repeated here under the new notational conventions, with both m and $n > 0$.

(55) (a) **... <a_m, [], []>... <a_1, [], []> <x, L, [y_1 R]> <y_1, L_1, [y_2 R_2]> ... <y_n, L_n, [b R_n]> <b, [a_1 ... a_m], R′>**

(b) **... <b, L′, [a_1 ... a_m]> <y_n, [b L_n], R_n> ... <y_1, [y_2 L_1], R_n> <x, [y_1 L], R> <a_1, [], []> ... <a_m, [], []>...**

If these configurations show up in a language, there is reason to believe that disharmony is involved. The occurrence of this type of strings is, however, not sufficient evidence that a language is disharmonious. The language should also have sentences in which the dislocated arguments are adjacent to the functor introducing them negatively, such that its argument string can be fulfilled by application. So, the configurations (55a) and (55b) should be accompanied by occurrences of canonical ordering, like (56a) and (56b), respectively.

(56) (a) **... <a_m, [], []> ... <a_m, [], []> <b, [a_1 ... a_m], R′> ...**

(b) **... <b, L′, [a_1 ... a_m]> <a_1, [], []> ... <a_m, [], []> ...**

If no such pairing occurs, an alternative lexical assignment would be possible under which the dislocated arguments $\mathbf{a}_i$ are introduced by the functor with head **x** in (55) and would fulfil their negative counterparts in that functor by application only. The complete diagnosis of disharmony as a minimal feature of a language therefore follows from the observation that a language has a set of functors the arguments of which positively occur both adjacent and non-adjacent to the functor introducing them negatively.

It cannot be a surprise that, from a minimalistic point of view, Dutch qualifies as a disharmonious language. At the level of its morphology, disharmonious configurations are analyzed in Moortgat (1988b). Morphological disharmony is not typical of Dutch, however. An English example is:

(57) discuss - ing the highlights.
vp/np vp\vp np

Suffixing the perfectly general *ing* to a verbal functor does not affect the verb's right valency. At the level of sentence grammar, it is the phenomenon of verb raising and cross serial dependencies that puts Dutch under suspicion of disharmony. In the next section I will elaborate on a minimal disharmonious account of this aspect of Dutch syntax. Before doing so, I will give an example of the way Disharmonious Composition operates in getting things straight. Theorem (58), in the antecedent of which one can observe dislocated arguments $\mathbf{a}_1$ and $\mathbf{a}_2$, is proven by derivation (59) and the Cut inference. Types of the form **<x,[],[]>** are abbreviated to **x** for readability.

(58) $\mathbf{a_2\ a_1 <t, L, [u_1\ R]> <u_1, L_1, [b\ R_1]> <b, [a_1\ a_2], R']> \rightarrow <t, [L_1\ L], [R'\ R_1\ R]>}$

(59) (a) $\mathbf{<u_1, L_1, [b\ R_1]> <b, [a_1\ a_2], R']> \rightarrow <u_1, [a_1\ a_2\ L_1], [R'\ R_1]>}$

(b) $\mathbf{<t, L, [u_1\ R]> <u_1, [a_1\ a_2\ L_1], [R'\ R_1]> \rightarrow <t, [a_1\ a_2\ L_1\ L], [R'\ R_1\ R]>}$

(c) $\mathbf{a_1 <t, [a_1\ a_2\ L_1\ L], [R'\ R_1\ R]> \rightarrow <t, [a_2\ L_1\ L], [R'\ R_1\ R]>}$

(d) $\mathbf{a_2 <t, [a_2\ L_1\ L], [R'\ R_1\ R]> \rightarrow <t, [L_1\ L], [R'\ R_1\ R]>}$

The derivational steps (59a) and (59b) are disharmonious since the left argument lists of the subordinate functors are not empty (cf. 53); the other two steps are applicative since both the argument list of the subordinate functors are empty. For the disharmonious operations there is an alternative path:

(60) (a) $\mathbf{<t, L, [u_1\ R]> <u_1, L_1, [b\ R_1]> \rightarrow <t,\ [L_1\ L], [b\ R_1\ R]>}$

(b) $\mathbf{<t, [L_1\ L], [b\ R_1\ R]> <b, [a_1\ a_2], R']> \rightarrow <t, [a_1\ a_2\ L_1\ L], [R'\ R_1\ R]>.}$

(60a) is disharmonious if $\mathbf{L_1}$ is not the empty string; (60b) is disharmonious by stipulation of the types $\mathbf{a_1}$ and $\mathbf{a_2}$ in the left argument list of the subordinate functor, just like (59b). The consequents of (59b) and (60b) are equal. Therefore, these two paths lead us to the same theorem:

(61) $\mathbf{<t, L, [u_1\ R]> <u_1, L_1, [b\ R_1]> <b, [a_1\ a_2], R']> \rightarrow <t, [a_1\ a_2\ L_1\ L], [R'\ R_1\ R]>.}$

The existence of alternative bracketings is due to the GDCR format (53). The example shows that this rule format for first order types abstracts from procedural aspects of reduction: no extrinsic order of reduction is implied. The rule has, however, no solution for a configuration like

(62) **a/b b\a**

<a′, L, [b′ Rrest]> <b′, [a′ Lrest], R>.

The GDCR is not able to decide which argument is to be cancelled. Although all kinds of procedural solutions may come into mind, I prefer to consider the occurrence of these configurations in a language to reveal that the language is beyond the limits of the restricted order variety the GDCR can cope with.

4. Dutch as a disharmonious language

Standard Dutch is said to be an SOV language (cf. Koster 1978), with verb clustering and crossing dependencies, that can be analyzed as a particular realization of verb raising (Evers 1975). In this approach, the wellformed (embedded) sentence (63) is related to the hidden configuration (64) by adjoining the verbs right- and upward to VP nodes.

(63) ... dat Jan Marie een afscheidskado wil geven.
that Jan Marie a farewell-gift wants give
`... *that Jan wants to give Marie a farewell gift*'

(64) ... [dat [Jan [Marie een afscheidskado geven] wil]]]

The crucial point is that at surface structure a string of arguments to verbs is to be licensed thematically by a string of verbs in such a way that e.g. the first argument is related to the first verb and the last argument to the last verb. The resulting network of relations R between arguments A and verbs V (65) is easily characterized as crossing dependencies produced by first-in-first-out stack handling of a **ww**-language.

(65) $A_1 \ldots A_i \ldots A_m \ldots V_1 \ldots V_j \ldots V_{j'} \ldots V_{j''} \ldots Vn$ $(n \leq m)$
$R(A_1, V_j), \ldots R(A_i, V_{j'}), \ldots R(A_m, V_{j''})$

Because this pattern shows up productively in Dutch, a conscientious grammar of the language is not context-free. The argument of Pullum and Gazdar (1982) to the contrary was based upon the idea that for purely syntactic purposes an indexed context-free grammar would suffice. Morphologically explicit case marking in Swiss German in the same type of networks (Haegeman and Van Riemsdijk 1986) and the recent resyntacticalization of thematic roles in linguistic theory make this point of view highly questionable, if not untenable, however.

Of course, there is a lot more to say about the ins and outs of the Dutch way of verb clustering. As a matter of fact, it was and is a main topic in the grammar of Dutch. Not all perspectives can be discussed here, however. Yet, some aspects of the clustering mode are highly relevant to its categorial characterization, to wit:

(a) the basic ordering of arguments
(b) the extent to which verb raising can be reduced to a lexical phenomenon
(c) the variation between verb raising and extraposition
(d) the nature of the raised phrase.

These four aspects are treated successively in the next paragraphs.

4.1. Basic order

The qualification of Dutch as an SOV language is misleading in so far as it suggests that all the arguments of a verb canonically occur to its left. This is

correct for nominal objects, but not for infinitival or sentential ones; in general, they cannot occur left of the verb that projects them:

(66) (a) *... dat Karel mij [te eten] overhaalde
that Karel me to eat persuaded
(b) *... dat de minister [met de honoraria te hebben
that the minister with the fees to have
gefraudeerd] ontkende
committed-fraud denied
(c) *Karel heeft nooit [dat hij kwam] beweerd.
Karel has never that he came claimed

In Dutch, either the complete verbal argument or a part including its head canonically occurs at the right of the selecting verb.

(67) (a) ... dat Karel mij overhaalde [te eten]
that Karel me persuaded to eat
(b) ... dat de minister ontkende [met de honoraria te hebben
that the minister denied with the fees to have
gefraudeerd]
committed-fraud
`... that the minister denied to have committed fraud with the fees'
(c) De minister heeft [niemand van zijn onschuld] kunnen
the minister has nobody of his innocence been-able
[overtuigen].
convince
`The minister has not been able to convince anyone of his innocence.'

Occasionally, a verb selecting infinitival or participial complements without overt complementizer *te* `to' may occur to the right of them, inducing the so-called green or German word order;

(68) (a) ... dat Anna [mij [programmeren] leren] wilde / [mij]
that Anna me program teach wanted / me
[programmeren] wilde [leren]
program wanted teach
`... that Anna wanted to teach me how to program'

(b) ... of de direktie hem [de p ost bezorgen] liet
whether the board him the mail deliver let
`... whether the board allowed him to deliver the mail'

(68) (c) ... dat men [hem de schuld in de schoenen geschoven] heeft
that one him the guilt in the shoes pushed has
`... that one put the blame on him'

This word order is marked, however, and subject to many restrictions. This also holds for the very exceptional cases of centre embedded propositional objects; they can occur left of their licensing verb, as a stylistic variation. Compare:

(69) (a) ?Karel heeft altijd dat hij gefraudeerd zou hebben ontkend.
(b) *Karel heeft dat Marie benoemd zou worden nooit beweerd.

It is remarkable that the verbs allowing their sentential objects — be it very marginally — to occur left, like *ontkennen* `deny', are those that make their sentential objects absolute *wh*-islands, whereas verbs like *beweren* `claim' are more liberal in this respect. Teun Hoekstra (personal communication) connects the variation in the degree to which sentential objects are rigid, to different ways of licensing: a Case regime on the absolute islands versus a tense regime on the peninsulas. This would underline my feelings that the sentential argument of a verb like **ontkennen** tends to be of a nominal nature, which would account for the remote possibility of occurring at the left of the verb, the canonical position for its nominal (i.e. Cased) arguments.

At the background of this succinct examination of Dutch verb clustering, we must realize that categorial grammar does not distinguish between underlying and derived configurations. From a linguistic point of view, it is basically an occurrence grammar. Moreover, a directional categorial grammar cannot supply the ordering variation of the type discusses here — arguments occurring to the left and the right of their functor — in a non-lexical way, as is captured by Steedman's invariancy postulate (33). In this spirit, we are well served by taking Dutch verbs essentially as functors looking for nominal arguments on their left and for verbal and sentential arguments on their right. Applied to Dutch, then, Steedman's Generalized Verb is any selection of arguments in the pattern

(70) **<vp, [NP PP NP], [vp s]>.**

Cross serial dependencies arise from disharmonious composition of these functors with another (subordinated) verbal functor, according to GDCR (53). Here is an example in the standard notation; the nominal arguments are indexed for easier recognition.

(71)

`... *that I don't want to help anybody teaching Mary to swim'*

	that I		*nobody*	*Marie*	*want*	*help*	*teach*	*swim*
...	dat	ik	niemand	Marie	wil	helpen	leren	zwemmen
	s/s	**np_1**	**np_2**	**np_3**	**$s\backslash np_1/vp$**	**$vp\backslash np_2/vp$**	**$vp\backslash np_3/vp$**	**vp**
							$vp\backslash np_3$	
						$vp\backslash np_2\backslash np_3$		
					$s\backslash np_1\backslash np_2\backslash np_3$			

(+ subsequent Application)

Note that whereas the GDCR does not imply any preference as to the order in which arguments are to be cancelled, it is necessary here to assume the finite verb to be of type **(s\np)/vp**. The same holds for the other verbal functors: they too must cancel their right arguments first. The bracketing has limited consequences under first order conditions, however: the functors body is still accessible for cancelling, as is made explicit in the GDCR format (53). Even within that assumption about the priority of rightward cancelling, there are alternative derivations that turn out to be equivalent; for example:

(72)

...	dat	ik	niemand	Marie	wil	helpen	leren	zwemmen
	s/s	**np_1**	**np_2**	**np_3**	**$s\backslash np_1/vp$**	**$vp\backslash np_2/vp$**	**$vp\backslash np_3/vp$**	**vp**
						$(s\backslash np_1\backslash np_2)/vp$		
							$(s\backslash np_1\backslash np_2\backslash np_3)/vp$	
							$s\backslash np_1\backslash np_2\backslash np_3$	

(+ subsequent Application)

The suggested ordering of verbal arguments — nominal arguments left, other arguments right — does not coincide with the almost traditional analysis of Dutch verb clustering as a product of such called Verb Raising. In the founding analysis of the topic, Evers (1975), the difference between Dutch and German sentence structure is explained as a necessary rightward move of infinitival arguments in the former language. This rightward move is either V-to-V adjunction, yielding cross serial dependencies, or extraposition. In both cases the complement originates at the left of its selecting verb. One peculiar thing about this generative analysis is the direction of the reconstruction; another is that the moves are obligatory for reasons independent of other principles of grammar. It is only very recently that one may observe tendencies to get rid of these particularities; unfortunately, the alternative analyses have not yet been published. Still, there is reason to believe that the mixed directionality of Dutch verbal functors, as expressed in (70), may turn out to be more than just a quest for descriptive adequacy within the limits of directed categorial grammar. Especially

the claim (implicit in the disharmonious approach to cross serial dependencies) that the nominal arguments of embedded verbs are dislocated — rather than the verbal part of an infinitival complement — is likely to survive the winds of change.

4.2. Disharmony

It is a mere fact of Dutch that the occurrence of cross serial dependencies is subject to restrictions. Two major classes can be distinguished under the hypothesis that in some way or another disharmonious composition is involved: restrictions on the main functor and restrictions on the subordinate functor. The subordinate functor is to be what Steedman (1990) calls a `generalized verb', i.e. a functor of type **vp\np|...** This means that not every directed functor qualifies for dislocation of its arguments, or alternatively: not every dislocation of arguments is an instance of `Verb Raising'. In Steedman (1990) and other work by the same author this kind of typological restriction on categorial rules is specified as part of a language's grammar, and seems to be motivated fairly well. Restrictions on the main functor are suggested by Moortgat (1988). There (p. 105) we find disharmonious composition modelled as a unary type transition for a certain class of lexical verbs: those `triggering' dislocation of nominal arguments. The impact of this restriction is given in (73).

(73) **x/y → (x\z)/(y\z)** if **x/y** is assigned to a verb in the class {*willen* `want', *moeten* `must', *kunnen* `can', ... }.

This way of limiting the scope of disharmony is misguided, I believe. The crucial point is that all kinds of adverbial and adsentential phrases should also be allowed to trigger or permit dislocation in order to get the correct structures. To see why, consider the following configuration:

(74) ... dat ik _ het boek _ aan Marie wil geven
that I the book to Marie want give

At least in the positions marked with `_' a whole class of modifiers could be added, e.g. *misschien* `maybe', *zonder de direktie daarover te informeren* `without informing the board on that matter', *uiterlijk maandag* `monday at the latest', *tijdens de receptie* `during the reception'. Each of them may be typed as taking sentences into sentences or VPs into VPs and as rightward looking for these arguments in the present example; their type will be **s/s** or **vp/vp**. Unless they occur at the left periphery, however, they will find to their right neither sentences nor VPs, but at best functors of type **x...\np,** with **x** being **s** or **vp**. The modifiers should be able to take over the lefthand arguments of this functor: otherwise, no derivation is possible. One is free to consider it as a derivative instance of a process that is got going by composition of verbal functors. Also one is free to describe the

transport either by composition or by a form of division, as in (73). What matters is that disharmonious power is called for by these modifiers. Since the class of phrases is by no means lexically closed in the same way as the class of verbal `Verb Raisers' is, disharmonious power — though not necessarily unlimited (cf. section 4.3) — must operate in the syntactic dimension. Ockham's razor and minimality, then, rule out particular provisions like Moortgat's (73).

4.3. Restricting disharmony

The fact that the class of phrases yielding a main functor in an instance of disharmonious composition is not lexically closed, does not mean that every phrase of a suitable type can be chosen as such. For example, many verbs with infinitival complements do not permit the nominal arguments of the embedded verb to be in dislocation, but require their complements to be completely saturated at their right. These verbs are called extraposing in the analysis of Evers (1975), mentioned above. The regime of these verbs is illustrated below by the example of *toegeven* `admit'.

(75) (a) ... dat hij niet wilde toegeven het kind te hebben ontvoerd
that he not wanted admit the child to have kidnapped
`... that he did not want to admit that he had kidnapped the child'
(b) *... dat hij niet het kind wilde toegeven te hebben ontvoerd

Given the notion of Generalized Directed Categorial Rule (53), there is an easy and natural way of recording these idiosyncrasies. Each argument of a functor may be indexed for the rule mode under which it is to be cancelled. The infinitival complement of *toegeven* `admit' and other extraposing verbs are indexed, then, in the type of those verbs for the most restrictive, i.e. applicative mode. The infinitival complement of `Verb Raisers' are indexed in the type of those verbs for the most liberal, disharmonious mode. This index is a property of an argument in a functional description, just like direction (cf. 33) is supposed to be by Steedman (1990). This analogy goes even further: while Flynn (1983) argued for parametrizing the directionality of arguments by their types, cancellation mode may be fixed for certain types too. For example, if the genuine NP is an island, the type **np** as an argument in a functional frame can be indexed for applicative rule mode in all assignments, thus blocking any cancellation mode under which arguments to an NP are transferred to a new type, in particular the disharmonious mode. This way of looking at rules and types is reminiscent of the effect of the modal operators introduced and discussed in Morrill (1988) and Hepple (1990) within the frame of a calculus. From now on, I will assume that arguments introduced negatively in a functor, are indexed `applicative only' (symbol $\owedge$) in the case that the argument type is not a domain for extraction or when it is a lexical property of the functor that its argument is to be consumed fully saturated.

This index blocks the GDCR if the argument lists of the intended subordinate functor are not empty:

(76) **<x, L, [⌂y R]> <y, L′, R′> → <x, [L′ L], [R′ R]>**
only if **[L′ L]** = **L** and **[R′ R]** = **R**.

Since we are dealing with minimal grammar and harmonious composition does not occur here, there is no need to relax the condition in (76) such that it applies to the list in one direction only. Furthermore, the index `applicative only' is on the negative argument and is therefore inherited by the consequent type under the GDCR if the argument is not cancelled, as is direction. One type lexically assigned to *toegeven* `admit' and *ontkennen* `deny' thus will be

(77) **<vp, [], [⌂vp]>**.

Two other regimes towards selected infinitival complements must be distinguished. First, there are the verbs that are indifferent with respect to dislocation of embedded nominal arguments. This regime is exemplified by *proberen* `try':

(78) (a) ... omdat zij [mij de Eiffeltoren] probeerde [te verkopen]
(b) ... omdat zij [mij] probeerde [de Eiffeltoren te verkopen]
(c) ... omdat zij probeerde [mij de Eiffeltoren te verkopen]

For these verbs, no additional provision is required. The type of *proberen* `try' will simply be **<vp, [], [vp]>**, and the GDCR operates unrestricted when the type is confronted with a functor headed by **vp**. Second, there are the verbs that used to be characterized as the real Verb Raisers. They force any nominal argument of an embedded verb to be in dislocation at their left. Among them we find the proper auxiliaries of time and modality, the passive auxiliary, semi-auxiliaries of all kinds and a verb like *willen* `want'. I will argue here that this regime can be expressed as a single requirement on the argument list in the disharmonious direction (i.e. the left arguments) of subordinate functors that the GDCR confronts those verbs with. An argument list that matches this requirement, is dubbed *virginal*. Virginity is defined below:

(79) An argument list **A** in a type **<a, A, A′>** is a virgin iff
either **<a, A, A′>** is a lexically assigned type
or **A** is a concatenation of virginal lists.

The first of the conditions gives the basic case. The second accounts recursively for virginity in types that are the yield of the GDCR, an instance of which is repeated here.

(80) **<x, L, [y R]> <y, L′, R′> → <x, [L′ L], [R′ R]>**.

The consequent type is not lexical, of course. Its argument list **[L′ L]** is virginal, however, if only both **L** and **L′** are virginal. The other of its argument lists, **[R′**

R], lost its virginity in action: at least **R** is not a list in a lexically assigned type. An argument list a member of which is cancelled by an instance of the GDCR, is no longer a virgin. It is therefore assumed that all lists start their career as a virgin, but on submission to the GDCR they may loose this property, depending on the direction in which that instance of the GDCR operates. In a generalized format, the GDCR is to handle virginity in the following way:

(81) Be **<x, L, R> <y, L′, R′> → <z, L″, R″>** the format of the GDCR.
L″ is a virgin if
L″ = [L′, L]
and
both **L** and **L′** are virgins, or
one of **L** and **L′** is a virgin and the other is the empty list,
and non virginal otherwise.
For **R″** the same condition holds, with **R″ = [R R′]**.

So, on entering the GDCR two virgin list yield virginity under concatenation if neither of them contains the argument that is cancelled by that instance of the rule. Otherwise, i.e. if they are the lists of the `active' direction, the concatenation is not a virgin. Clearly, one of the concatenation results **L″** and **R″** cannot maintain virginity. An argument list that is empty by assignment or emptied as a result of the GDCR, does not block the virginity of a concatenation if it is not in the active direction. Such is the logic of virginity. This may be seen as a way of implementing the observation by Houtman (1984) that some types are sensitive to the `lexicality' of arguments, in the dynamics of the present categorial setting.

Now suppose that the verbs that trigger dislocation of embedded nominal arguments, have their infinitival complement argument marked for virginity of its left argument list (symbol ▲). This argument can only be cancelled, then, if the subordinate functor has a virgin list in the only direction in which virginity can be maintained. Its effect on the GDCR will be:

(82) (a) **<x, L, [▲y R]> <y, L′, R′> → <x, [L′ L], [R′ R]>**
only if **L′** is virginal
(b) **<y, L′, R′> <x, [▲y L], R> → <x, [L′ L], [R′ R]>**
only if **R′** is virginal

Willen `want', for example, would thus receive the lexical assignment **<vp, [], [▲vp]>**. Then, the constructions (83) are properly accounted for. The type of the NPs is simplified, since their internal structure does not contribute anything of relevance here. The virginity of argument lists, according to definition (81), is indicated by a cap (^) behind the list.

(83) (a)

	that I		*want*	*sleep*
	...dat	ik	wil	slapen.
		np	**<s,[np]^,[▲vp]^>**	**<vp,[]^,[]^>**

<s,[np]^,[]>

The argument lists of *slapen* `sleep' are virginal by assignment. Hence, the cancelling of **▲vp** is fine, although the rule mode is applicative rather than compositional.

(b)

that I	*a book*	*want*	*read*
...dat ik	een boek	wil	lezen
np	**np**	**<s,[np]^,[▲vp]^>**	**<vp,[np]^,[]^>**
		<s,[np np]^,[]>	

Again, the left argument list of *lezen* `read' is virginal according to (81). But now the rule mode is disharmoniously compositional: the characteristic case.

(c) ... dat ik hem het pamflet wil proberen te overhandigen
that I him the pamphlet want try to hand-over
`... that I want to try to hand the pamphlet over to him'

(i)

... dat	ik	hem	het pamflet		
	np	**np**	**np**		
wil			proberen	te overhandigen	
<s,[np]^,[▲vp]^>			**<vp,[]^,[vp]^>**	**<vp,[np np]^,[]^>**	
			<vp,[np np]^,[]>		
		<s,[np np np]^,[]>			

(ii)

... dat	ik	hem	het pamflet		
	np	**np**	**np**		
wil			proberen	te overhandigen	
<s,[np]^,[▲vp]^>			**<vp,[]^,[vp]^>**	**<vp,[np np]^,[]^>**	
		<s,[np]^,[vp]>			
		<s,[np np np]^,[]>			

In these examples the difference between infinitives with and without *te* `to' is neglected. Yet, it is obvious that the dislocation of the embedded nominal arguments proceeds smoothly under either analysis. In both cases the virginity requirements for *willen*'s `want' **vp** argument are met. Note that in (83cii) neither the right argument list of *willen proberen* `want to try' is virginal nor is its **vp** argument marked for virginity; in accordance with the GDCR, the right argument looked for by this constituent originates from the type of *proberen* `try', which has no such marking. Here is a variation on this theme.

(d)

... dat	ik	hem			
	np	**np**			
wil		proberen	het pamflet	te overhandigen	
<s,[np]^,[▲vp]^>		**<vp,[]^,[vp]^>**	**np**	**<vp,[np np]^,[]^>**	

<s,[np]^,[vp]> **<vp,[np],[]^>**

<s,[np np],[]>

In this example only one nominal argument of *overhandigen* `hand over' is dislocated. The acceptability of this sentence is due to the afore-mentioned indifference of *proberen* `try' with respect to disharmony. Notwithstanding the fact that the left argument list of *het pamflet te overhandigen* `to hand the pamphlet over' is no longer virginal, it is accepted as a subordinate functor to the type of *willen proberen* `want to try', since this type does not require virginity. The type that does require this property — the type of *willen* — is satisfied properly in this respect by *proberen*. No alternative analysis is possible here: the constituent *proberen het pamflet te overhandigen* `try to hand the pamphlet over' can be formed by the GDCR, but it cannot hand a virginal left argument list to *willen*. I will elaborate on this topic in section 4.4.4. The next example is shows the normal structural ambiguity, but only one analysis is demonstrated.

(e) ... dat ik

np

wil proberen hem het pamflet te overhandigen

<s,[np]^,[▲vp]^> **<vp,[]^,[vp]^>** **np** **np** **<vp,[np np]^,[]^>**

<vp,[],[]^>

<vp,[]^,[]>

<s,[np]^,[]>

No argument is in dislocation here. The type of *proberen* `try' is consistent with a saturated complement. The left argument list of *proberen hem het pamflet te overhandigen* `try to hand the pamphlet over to him' is the concatenation of an empty list and an (also empty) virgin list, which is a virgin again by definition (81). Thus, the requirement on the complement of *willen* `want' is met. Next, there are two slightly more complicated cases with an embedding of the verb *dwingen* `force'.

(f) ... dat ik hem wil dwingen Piet te ontslaan
that I him want force Piet to fire
`*... that I want to force him to fire Piet.*'

(i) ... dat ik hem

np np

wil dwingen Piet te ontslaan

<s,[np]^,[▲vp]^> **<vp,[np]^,[⌂vp]^>** **np** **<vp,[np]^,[]^>**

<vp,[],[]^>

<vp,[np]^,[]>

<s,[np np]^,[]>

(ii) * ... dat ik

		np			
wil	hem	dwingen	Piet	te ontslaan	
<s,[np]^,[▲vp]^>	**np**	**<vp,[np]^,[⌂vp]^>**	**np**	**<vp,[np]^,[]^>**	
				<vp,[],[]^>	
			<vp,[np]^,[]>		
		<vp,[],[]^>			

(▲ requirement not met)

In (83fi) everything goes well. *Dwingen* `force' is satisfied in its right argument by an application mode of the GDCR, as required. The resulting type has a virginal left argument list. This is accepted by *willen* `want'. Consequently, the nominal argument of *dwingen*, but not the nominal argument of *ontslaan* `fire', is dislocated. In example (83fii), however, the nominal argument of dwingen is consumed by *dwingen Piet te ontslaan* `force to fire Piet'. The resulting type cannot hand a virginal left argument list to *willen*. Hence, the derivation is blocked. Because there is no alternative analysis, the sentence is correctly ruled out. Here are, finally, two more examples of derivations that are correctly blocked for the very same reason: under no analysis can *willen* be satisfied by a functor with a virginal left argument list.

(g)

(i) *... dat ik wil hem het pamflet overhandigen

np	**<s,[np]^,[▲vp]^>**	**np**		**np**	**<vp,[np np]^,[]^>**
			<vp,[],[]^>		

(ii) *... dat ik hem wil het pamflet overhandigen

np	**np**	**<s,[np]^,[▲vp]^>**		**np**	**<vp,[np np]^,[]^>**
					<vp,[np],[]^>

It is apparent from these examples that the proposed restrictions on the way of operation of the GDCR — i.e. restrictions on the nature of the subordinated functor, encoded in type assignment as properties of negative types — in combination with the notion of virginity of argument lists supply a dynamic account of major idiosyncracies of Dutch sentence structure. This account can be summarized as the assignment of one of the following types to verbs that select verbal complements.

(84) (a) **<vp,L^,[▲vp]^>** (example: *willen* `want', with **L = []**)
(b) **<vp,L^,[⌂vp]^>** (example: *ontkennen* `deny')
(c) **<vp,L^,[vp]^>** (example: *proberen* `try', with **L = []**).

In the next paragraph some further aspects of this approach are discussed.

4.4. The typology of disharmony

4.4.1. Verbs

Up to now, no attention was payed to the precise way of deriving the type of finite verbs. It was tacitly assumed that a finite verb introduces the sentential type **s**, and therefore is to be considered as the main functor in a sentence, from a syntactic point of view. Of course, there is a lot more to say about this. Ades and Steedman (1982), for example, argue that it is adequate for Dutch to consider the subject the sentence's main functor. Such a strategy is consistent with higher-order categorial grammar and the semantic program of generalized quantification, founded in the work of Richard Montague and in Barwise and Cooper (1981). Discussing all the implications hereof, however, is far beyond the limits of this study. The characterization of finite verbs as functors with head **s** is merely meant to distinguish their combinatorial role in the sentence from the role of infinitival forms. And even this does not imply that infinitival forms could not introduce sentential types; Cremers (1983a) favors some distinctions with regard to the categorial nature of infinitival phrases. Yet, the considerations thus far enable us to give a concise characteristic of the different roles that Dutch finite verbs may fulfil. In particular, the different assignments of types that permit the finite verb to occur in final as well in first or second position in a sentence are straightforward.
Given an initial assignment of a general verbal type (85), to each finite instance of a verb each of the types in (86) can be assigned lexically.

(85) **<vp, L^, R^>**

(86) (a) **<s, [L np]^, R^>** (verb final)
(b) **<s, [np]^, [L ⌂R]^>** (verb second)
(c) **<s, []^, [np L ⌂R]^>** (verb first)

In each finite type, a **np** is added as the subject argument. Its position in the argument lists characterizes the combinatorial options, in combination with a restructuring of the original argument lists and an `application only' marking of all the right arguments in two of the three finite types. In the case that a right argument is marked for virginity, this property is supposed to be overruled by the general applicative requirement. This accounts for the fact that finite verbs in final or second position do not support crossing dependencies. Thus, a perfectly

general assignment of types to finite verbs is provided in the lexicon; the sample (86) is not the only option in this respect.

4.4.2. Non-verbal functors

In section 4.2 it was argued that disharmony is needed in the syntax of Dutch to account for the role of adverbial functors in the dislocation of nominal arguments. In fact, they enforce the same liberal regime on their right argument as verbs of the *proberen* `try' type do; under the present conventions they would simply be typed **s/s** or **vp/vp**. There is one peculiar aspect of their behaviour, however, that should be addressed here: they cannot occur in the right domain of a verb that triggers dislocation — a `real Verb Raiser' — but they are allowed in the right domain of other verbs that take infinitival complements. Here are some data with the adverbial *zo snel mogelijk* `as soon as possible'.

(87) (a) ... dat ik zo snel mogelijk hem het boek wil bezorgen
that I as soon as-possible him the book want deliver
`*... that I want to deliver the book to him as soon as possible*'
(b) ... dat ik hem zo snel mogelijk het boek wil bezorgen
(c) ... dat ik hem het boek zo snel mogelijk wil bezorgen
(d) *.. dat ik hem het boek wil zo snel mogelijk bezorgen
(e) Ik wil hem zo snel mogelijk het boek bezorgen.
(f) ?... dat ik hem het boek wil proberen zo snel mogelijk te bezorgen
(g) ... dat ik wil proberen zo snel mogelijk te komen
that I want try as soon as-possible to come

(88) (a) ... dat ik zo snel mogelijk hem het boek probeer te bezorgen
that I as soon as-possible him the book try to deliver
`*... that I try to deliver the book to him as soon as possible*'
(b) ... dat ik hem het boek zo snel mogelijk probeer te bezorgen
(c) ... dat ik hem het boek probeer zo snel mogelijk te bezorgen

I have no explanation for the ungrammaticality of (87d) and the questionability of (87f). But on scrutinizing the data we can find a description of the adverbial's behaviour that is consistent with each of them. For this sake, assume that the adverbial affects the virginity of all the argument list of a subordinate functor that it is confronted with by the GDCR, i.e., assume that its right argument is marked in such a way that the GDCR with the adverbial as the main functor is instantiated as:

(89) **<vp, []^, [vp]> <vp, L^, R^> → <vp, L, R>**

Although the left argument lists are not activated and both are virginal, the left argument list of the consequent is deflowered. If so, the consequent does not qualify for cancelling a negative type **▲vp**, occurring in the right argument list of

e.g. *willen* `want'. Such a property of the adverbial's type identifies correctly the examples (87d) and (87f) as the only sentences in the set to which it is harmful. In all other sentences, as one may easily check, the type that looses its virginity upon the confrontation with the adverbial, is not confronted with a type that requires virginity. Particulary interesting in this regard is the opposition between (87f) and (87g). In the first sentence, under one analysis, the constituent *zo snel mogelijk te bezorgen* `deliver as soon as possible' hands a nonvirginal nonempty left argument list to *proberen* `try': virginity cannot be restored, however, according to (81). There is another analysis here, as in previous cases, in which the constituent *willen proberen* `want to try' shows up. This constituent does not insist on virginity and accepts *zo snel mogelijk te bezorgen* as a candidate for cancelling its right argument. This case is on a par with the situation in (83d). Sentence (87g) is fine under either analysis: in the bracketing *[wil proberen][zo snel mogelijk te komen]* `want to try to come as soon as possible' virginity of the left argument list of the second constituent is not required. In the bracketing *[wil][proberen [zo snel mogelijk te komen]]* the seemingly deflowered left argument list of *zo snel mogelijk te komen* `to come as soon as possible' is empty, and does not resist virginity of the empty concatenation that is produced in the formation of the second constituent (cf. definition 81). Consequently, the cancelling requirement of *wil* `want' is met. The grammaticality of (87e) is due to the fact, mentioned in the preceding section, that finite verbs in second position are `applicative only' (cf. 86b).

The conclusion must be, then, that the particularities of the distribution of nonverbal functors regarding disharmony can be expressed quite smoothly in the present set up for handling restricted permutation by disharmony.

4.4.3. Complex predicates

Two other intrinsicalities of Dutch word order also appear to be connected to the `way of being' of argument lists in the GDCR: the close relation between resultative predicates like *groen* `green' and the verbal complex in (90), and the liberal distribution of particles like *weg* `away' in (91).

(90) (a) *... dat ik het hek met de roller wil groen verven
that I the fence with the roller want green paint
(b) ... dat ik het hek met de roller groen wil verven
(c) *... dat ik het hek groen met de roller wil verven

(91) (a) ... dat ik het goud weg wil proberen te brengen
that I the gold away want try to bring
`... that I want to try to bring the gold away'
(b) ... dat ik het goud wil weg proberen te brengen

(c) ... dat ik het goud wil proberen weg te brengen
(d) *... dat ik het goud weg zo snel mogelijk wil proberen
that I the gold away as soon as-possible want try
te brengen
to bring
`... that I want to try and bring the gold away as soon as possible'

For all kinds of reasons one may want to subsume these items under one class. I will concentrate on distributional properties, however, and from that point of view there is a subtle difference between *groen* `green' and *weg* `away'. What they have in common can be read from the ungrammaticality of (90c) and (91d): they block intervention of adverbial modifiers between them and the verbal function they are bound to be consumed by. Seen as arguments of the deepest embedded verb, both *groen* and *weg* obviously require that verbal functor to be virginal in the relevant direction. In other words, if we type them as **pred**, they can be said to insist on the presence of the cap at **[pred L]^** in the following instance of the GDCR:

(92) **<pred, []^, []^> <vp, [pred L]^, R> → <vp, L, R>**

If we assume that **pred** is such that it imposes this requirement on the antecedent's main functor, the ungrammaticality of (90c) and (91d) is a direct consequence of the — independently motivated — behaviour of adverbials expressed in (89): the adverbial does enhance dislocation of the left arguments of the verbal complex, but only by destroying its virginal status. At the confrontation of the new constituent with the **pred** type, it is not able to match the cap condition of (92). The distributional difference between *groen* and *weg* can be read from the opposition between (90a) and (91a). *Groen* acts as a `normal' argument: when it is consumed (by application), the resulting new argument list is no longer virginal, and not able to match the negative **▲vp** in the type of *wil* `want' that requires virginity. Hence, (90a) is out, as expected. *Weg* `away', on the other hand, does not seem to have the same effect. In the line pursued here, this means that this particle, when consumed, does not affect the virginal status of the main functor in the relevant direction. This is expressed in the following instance of the GDCR; watch the cap at **L** in the consequent.

(93) **<predpart, []^, []^> <vp, [predpart L]^, R> → <vp, L^, R>**

So, a verbal functor having cancelled its **predpart** argument, is still available for cancelling a negative **▲vp** type, as in (91b) and (91c). This treatment of the difference between *groen* and *weg* has a flavour of naturalness, I would say. *Weg* is more or less incorporated in a verbal complex as a lexical part of it, and as such its being consumed does not affect the lexically rooted property of virginity.

4.4.4. The third construction

Finally, let us return to a topic that was broached in the presentation of disharmonious effects in section 4.3., example (83d). Under the approach of cross-serial dependencies advocated here, the existence of the so called `third construction' is more or less predicted. This construction is intermediate between the cancellation by application of fully saturated VPs — the classical Extraposition case — and the cancellation by disharmony of virginal VPs, which is traditionally analyzed as Verb Raising. It is discussed in some detail by Den Besten and Rutten (1989) and Mulder (1987). Two examples of the third construction are:

(94) (a) ... toen hij [mij] probeerde [m'n eigen auto te verkopen]
when he me tried my own car to sell
`...when he tried to sell my own car to me'
(b) ... dat hij [de chef] meende [voor zich te hebben gewonnen]
that he the boss thought for himself to have won
`... that he thought to have won the boss' sympathy'

To the right of the finite verb occurs a phrase that is neither lexical nor virginal: one nominal argument of *verkopen* `sell' is dislocated, the other is not. It is important to note that this way of complementation is only possible with verbs that allow, in the traditional approach, both extraposition and V-raising of their infinitival complement. In the present setting these are the verbs, like *proberen* `try' in (84), that have no restrictions on the argument lists of a subordinate functor against which its complement type is cancelled. But if so, they will tolerate any argument list, including non-empty non-virginal ones like those of *m'n eigen auto te verkopen* `sell my own car' and *voor zich te hebben gewonnen* `to have won sympathy'. The relevant part of the derivation of (94a) is given below.

(95) (a) probeerde: **<s, [np]^, [vp]^>**
hij, mij, m'n eigen auto: **<np,[]^,[]^>** (short: **np**)
te: **<vp, []^,[▲vp]>**
verkopen: **<vp, [np np]^ ,[]^>**
(b) [te verkopen]:
<vp, []^, [▲vp]^> <vp, [np np]^, []^> → <vp, [np np]^, []>
(c) [m'n eigen auto te verkopen]:
np <vp, [np np]^, []> → <vp, [np], []^>
(d) probeerde m'n eigen auto te verkopen:
<s, [np]^, [vp]^> <vp, [np], []^> → <s, [np np], []>
(e) mij probeerde m'n eigen auto te verkopen:
np <s, [np np], []> → <s, [np], []^>.

If the negative **vp** argument of *probeerde* `tried' in (95d) would have been indexed ⌂ or ▲, no cancelling could be executed, since the left argument list of the intended subordinate functor is neither empty nor virginal. Since we have independent reasons for not restricting the way the **vp** argument of *proberen* is

cancelled in either of these two ways, the fact that exactly the *proberen* class of verbs allows for the `third' construction supports the description of V-raising phenomena in terms of conditions on the mode of GDCR.

Although many idiosyncrasies of Dutch word order remain undiscussed here, the survey in these sections has shown that a minimalistic categorial approach in which disharmony is the major source of discontinuity, may adequately extend to a proper grammar of Dutch. On this basis, chapter 3 will present a computational treatment of coordination at the background of a grammar restricted in this sense. In between, chapter 2 addresses the syntactic nature of coordination in a parsing perspective.

5. Summary

In this chapter, the notion of minimal categorial grammar for a language was introduced. The Lambek calculus was argued to be unsuitable for a minimalistic approach to language description. Disharmonious composition of functors turned out to be the least permutational power that can be added to an elementary (AB) grammar, whereas harmonious operations only add to the strong recognizing force of a grammar and can be dispensed with in a minimalistic program. Disharmony was argued to be a major syntactic aspect of the grammar of Dutch. Its operation is restricted by general and lexical specifications affecting the generalized cancellation recipe given by (53). In this rule important invariances with respect to directionality and arithmetical balances are preserved.