1.1.2.1. Formalizing the syntax

1.1.2.1.1. BASIC SETS: TYPES, MODALITIES AND CATEGORIES

The syntax lives on three well-defined sets:

(1) a finite set Types of **types**
(2) a finite set Mods of **modalities**
(3) a set Cats of **categories**

The set Types of types consists of literals naming syntactic and semantic equivalence classes over lexical phrases. The only restriction imposed on this set is finiteness. At least one of the types is marked for sentencehood. One may reduce the number of sentential types to one, but language calls for telling propositions from questions and imperatives. A typical set of types may consists of literals like **s** for propositional sentences, **q** for questions, **np** for names and nominal quantifiers, **vp** and **ip** for tensed and untensed verb phrases, and so on. Clearly, the distribution of types over the grammar expresses our insight in the phrasal tiers but the types themselves lack any formal content.

The set Mods of modalities is also taken to be a set of literals disjunct from Types. Modalities stand for modes of merging categories. They are defined pointwise. Since the number of merge modes will turn out to be finite, Mods is finite too.

The set Cats of categories holds the basic data structures of the grammar. Every category is a triple <Head, LeftArguments, RightArguments>. The head is a single, bare member of Types. LeftArguments consists of a pair <Flag, LeftArgumentList>; Flag is an integer, holding information as to the present state of LeftArgumentList. RightArguments is constructed accordingly. Both the left and the right argument list consist of a finite number - possibly zero - of pairs <Type, Modality> in Types×Mods. The left arity of a category is the number of pairs <Type, Modality> in LeftArgumentList. Equally, the right arity of a category is the number of modalized types in the RightArgumentList. The arity of the category plain is the sum of its left and right arity. Basically, a category is to be seen as an *n*-place function mapping categories onto categories, where *n* reflects its arity.
For notational convenience, categories are written in the format

(4) Head\ LeftFlag~LeftArgumentList / RightFlag~RightArgumentList.

Non-empty argument lists are written [*Type1^Mode1, Type2^Mode2, ...]* or *[Type1^Mode2|Rest].*
This format of categories is reducible to the flat types of Buszkowski (1988). The only element added is the treatment of all the argument at a certain side of the head as one object, a list or stack.

Every phrase in the language that is to be defined is assigned to one or more categories, by lexicon or by grammar. Here is an example from Dutch.
The article *de* ('the') is lexically assigned to the category **np\0~[]/0~[n^i]**. Its left arity is zero; its right arity is one. The category expresses that its phrases are of type **np** if combined with phrases of type **n**. To combine with a phrase is to merge with a category to which that phrase is assigned. It will operate according to merge mode *i* on a category headed by *n.* If mode *i* allows for operating on *e.g.*. **n\0~[]/0~[]** and the noun *duivelsuitdrijver* ('exorcist') is lexically assigned to that category, the grammar may assign the phrase *de duivelsuitdrijver* to the category **np\ 0~[]/ 1~[]**. The modalized argument *n* in the original category for *de* has been cancelled, and RightFlag adapted. The category thus specifies the nature of categories to be operated on by listing heads in the argument lists. The associated modalities convey the

conditions under which this merge may unfold in the modalities. Furthermore the category stores concise information as to its merge history in the flags at the argument lists. Merging categories is therefore a complex, asymmetric operation.

Merging categories is the only combinatory operation in the syntax. It was originally introduced in Cremers (1993). The operation is a generalization of Generalized Composition for Combinatory Categorial Grammar, as it was framed by Joshi *et al.* (1991) to capture the grammar engines constructed by Steedman (1996, *e.g.*).
In Combinatory Categorial Grammar, Generalized Composistion is the main engine of analysis; here are its two instances as presented by Joshi *et.al.* (1991).

(5) $x/y \quad (\ldots(y|z_1)|z_2 \ldots |z_n) \Rightarrow (\ldots(x|z_1)|z_2 \ldots |z_n)$
(6) $(\ldots(y|z_1)|z_2 \ldots |z_n) \quad x\backslash y \Rightarrow (\ldots(x|z_1)|z_2 \ldots |z_n)$

x/y and $x\backslash y$ are considered to be the primary category of the compositions (5) and (6), respectively; the other one at the left of $\Rightarrow$, headed by y, is called the secondary category. Every occurrence of $|z_i$ is a unit, representing either $\backslash z_i$ or $/z_i$. Directionality of arguments is not affected by composition (Steedman 1990). Generalized Composition cancels the argument $/y$ in the primary category against the secondary category's head y and yields the remaining structure x of the primary category with all the arguments of the secondary category stacked on top of it.
Categorial List Grammar restricts Generalized Composition by requiring the cancelled type y to be primitive or atomic, i.e. without internal structure. This restriction is referred to as *linearity*. In the terminology of König (1990), the resulting grammars would be characterized as first order grammars, since there is only one relevant level of type embedding. A type is either a head or an argument to that head. Hepple (1996) describes a linearization procedure as a compilation of higher order categorial grammars for parsing. Moreover, the syntax presented here will take into consideration the full internal structure of type x, which is not specified in Generalized Composition. As a consequence, Categorial List Grammar extends generalized composition form (5) to two different operations:

(7) $(\ldots(p|w_1)|w_2\ldots|w_m)/y \quad (\ldots(y|z_1)|z_2 \ldots |z_n) \Rightarrow$
$(\ldots(\ldots(p|z_1)|z_2 \ldots |z_n)|w_1|w_2\ldots|w_m)$
(8) $(\ldots(p|w_1)|w_2\ldots|w_m)/y \quad (\ldots(y|z_1)|z_2 \ldots |z_n) \Rightarrow$
$(\ldots(\ldots(p|w_1|w_2\ldots|w_m)|z_1|z_2 \ldots |z_n)$

From these, the second is a more explicit version of (5). The first form of composition is not covered by Generalized Composition, because there the primary category's head and arguments can not be seperated. Furthermore, Categorial List Grammar will split up both the secondary and the primary category as to the directionality of arguments. CLG then collects the set of arguments in each direction. Consequently, the extension will be split up again in four different modes of composition. They are solely distinctive as to the relative orderings of the directed arguments in the consequent with respect to their source (graphical marking is just for convenience):

(9) $\mathbf{p}\backslash \mathit{pl}_1\ldots\backslash \mathit{pl}_n\ /\mathrm{pr}_1\ldots/\mathrm{pr}_m/\mathbf{s} \quad \mathbf{s}\backslash\mathrm{sl}_1\ldots\backslash\mathrm{sl}_k\ /\mathit{sr}_1\ldots/\mathit{sr}_l \Rightarrow$
$\mathbf{p}\backslash\mathrm{sl}_1\ldots\backslash\mathrm{sl}_k\ \backslash \mathit{pl}_1\ldots\backslash \mathit{pl}_n/\mathit{sr}_1\ldots/\mathit{sr}_l/\mathrm{pr}_2\ldots/\mathrm{pr}_m$
(10) $\mathbf{p}\backslash\mathrm{sl}_1\ldots\backslash\mathrm{sl}_k\backslash \mathit{pl}_1\ldots\backslash \mathit{pl}_n/\mathrm{pr}_1\ldots/\mathrm{pr}_m/\mathit{sr}_1\ldots/\mathit{sr}_l$
(11) $\mathbf{p}\backslash\ \mathrm{pl}_1\ldots\backslash\mathrm{pl}_n\backslash\mathrm{sl}_1\ldots\backslash\mathrm{sl}_k/\mathrm{sr}_1\ldots/\mathrm{sr}_l/\mathrm{pr}_1\ldots/\mathrm{pr}_m$
(12) $\mathbf{p}\backslash\ \mathit{pl}_1\ldots\backslash \mathit{pl}_n\backslash\mathrm{sl}_1\ldots\backslash\mathrm{sl}_k/\mathrm{pr}_1\ldots/\mathrm{pr}_m/\mathit{sr}_1\ldots/\mathit{sr}_l$

By simple computation, CLG extends every instance of Generalized Composition to eight patterns. The properties of these patterns are discussed in the remainder of this section. In the formatting of CLG below the directed arguments of a category are bundled into two single objects, ordered lists or stacks. In order to keep composition or merge of categories functional, a finite number of distinct modes of composition will be distinguished.

1.1.2.1.2. MERGE

1.1.2.1.2.1. Basic properties

Merge basically sends pairs of categories to categories, and therefore lives in the functional space $C^{C \times C}$. It consists of four suboperations:

(13) cancelling a type in a pair <Type, Mode> at the top of an argument list of one category against the head of the other
(14) appending two pairs of argument lists
(15) reflagging the resulting argument list
(16) constructing a new category from the components

The whole operation is subjected to one out of a finite set of modalities, the one that is associated with the type to be cancelled.

Here is an example of a merge of two categories C1 = *s\0~[pp^isl]/0~[vp^rais***]** and C2 = *vp\0~[np^isl]/1~[vp^cons]*. The only possible cancellation involves the type *vp* in the righthand argument list of C1 and the *vp* head of C2 , under modality *^rais*. The cancellation can be effected if C2 is the right member of the pair of input categories and the internal structure of C1 and C2 complies with the constraints defined by *^rais*. Suppose that both conditions are met. If we indicate the asymmetric merge with the infix ⊗, the merge could look like

(17) C1 ⊗ C2 → C3
s\0~[pp^isl]/0~[vp^rais] ⊗ vp\0~[np^isl]/0~[vp^cons] →
s\0~[np^isl, pp^isl]/1~[vp^cons]

In this merge, the left argument lists are appended in such a way that the argument list delivered by C2 is prefixed to the left argument list stemming from C1. Herewith an order of future cancellation is fixed. At the right hand side only the argument list of C2 survives, as the right argument list of C1 is emptied after cancellation of its only argument. The flag *1* at the new list indicates that it was rightward cancelling that brought this merge about: the right hand argument list is the affected one.

Reversing the point of view, one may now define what exactly are the constraints imposed by the *^rais* mode. In fact, two different patterns have to be stated, depending on the direction of the cancelling. The particular merge mode imposed by *^rais* is indicated by the modalized merge operator $\otimes_{rais}$. As merge is essentially asymmetric, it is appropriate to discriminate systematically between the two input categories. The category that has an argument at the top of one of its lists cancelled, is dubbed the *primary* category. Its head is *Prim*, and all its others components are preceded by a *p*. The other components are identified by *f* and *a* for flags and argument list, respectively, and by *r* and *l*, for left and right side, respectively. The other category is deemed to be the *secondary* one, with head *Sec* and *s*'s in stead of *p*'s. Finally, asymmetric append is marked by ⊕. Here then are two possible instances of the merge mode *^rais*. Capital onsets indicate variables.

(18) ($\otimes_{rais}$ /)
Prim\Plf~Pla/Prf~[Sec^rais|Pra] $\otimes_{rais\_}$ Sec\Slf~Sla/Srf~Sra →Prim\Slf~Sla⊕Pla/1~Pra⊕Sra

(19) ($\otimes_{rais}$ \)
Sec\0~Sla/Srf~Sra $\otimes_{rais}$ Prim\Plf~[Sec^rais|Pla]/Prf~[] $\rightarrow$Prim\1~Pla$\oplus$Sla/Srf~Sra

The first of these two merge modes does not impose any constraint on the input categories apart from the presence of cancelable types. As for the output, it specifies prefixing of the secondary argument list at the left hand side and the reverse at the right hand side. The flag *1* at the right hand side marks affectedness of this argument lists. The flag at the other side is provided by the secondary category.
Unlike ($\otimes_{rais}$ /), ($\otimes_{rais}$ \) does impose restrictions on the structure of the input categories. It requires the left argument list of the secondary category to be hitherto unaffected, and specifies emptiness for the right argument list of the primary category. In its output specifications it essentially behaves like its rightward dual, *modulo* directionality.

Here is a sample instantiation of ($\otimes_{rais}$ /). The Dutch verb *willen* ('want') is a typical verb raiser. It selects, among other, infinitival complements and adjoins to the left of their verbal head. Consider one of its finite forms, for example the singular past tense form for embedded, *i.e.* verb final sentences: *wou*. This form introduces a subject argument with which it agrees to its left, and an infinitival complement to its right (see part 2 for a discussion of this directionality). It would typically be assigned to the category *s_emb\0~[np]/0~[vp^rais]*. Both lists are marked zero, as the category originates from lexical assignment. Take furthermore the verb *gaan* ('go'). One of its combinatorial options is to create a vp by selecting a directional pp to its left. Thus, it will be assigned lexically to the category *vp\0~[ppdir]/0~[]*. The merge mode ($\otimes_{rais}$ /) specified above yields the following composition to these two categories, deriving a new category for the string *wou gaan* ('wanted go'):

(20) s_emb\0~[np]/0~[vp^rais] $\otimes_{rais}$ vp\0~[ppdir]/0~[] $\rightarrow$
s_emb\0~[ppdir, np]/1~[]

In appliance with the format for $\otimes_{rais}$ the left argument list of the secondary catgory, here consisting only of the argument *ppdir*, is appended as a prefix to the left argument list of the primary category. By implication, the resulting category will have to cancel *ppdir* before *np*. This reflects the state-of-affairs that the subject of *wou gaan* is more pheripheral to that phrase than its directional argument.

1.1.2.1.2.2. Finiteness of merge and modalities

It is by no means necessary that every modality be specified both for rightward and for leftward cancellation. Given the nature of the example (20) for ($\otimes_{rais}$ /), it is even unlikely that the grammar of Dutch would give rise to ($\otimes_{rais}$ \). Merge is inevitably asymmetric. One head is cancelled, the other head is persistent. The argument lists in one direction are unaffected, those in the other direction loose an argument. For each direction, the unfolding or execution of one argument list will be suspended, while the other is readily available for cancellation. Since linearity matters in natural language, directional duals for merging modes will be the exception, rather than the rule.
Plural specifications in one direction are possible. They should address disjunct subdomains of Cats × Cats, though, as merge is functional. Nevertheless, it can be proven that the set of expressible merges and therewith the set of merge modes is finite. To see why, consider first this characterization of an expressible merge. A merge aims at the cancellation of one exactly type at the top of one argument list and does not refer to other types in that or other argument lists. It specifies the heads of two categories and exactly two appends of argument lists. It specifies at most two flags at these lists. Therefore, the gamma of specifications by a merge mode is rather limited. Moreover, all the components of a merge come from limited sets or classes of suboperations. First, the set Types is taken to be finite, in accordance with the standard condition in formal grammar that both the terminal and the nonterminal alphabet be so.

As for input conditions, merge modes may specify of each of four argument list that they are empty or nonempty, and may specify flags for these lists. The number of different flags is very limited, by definition. As for output conditions, a merge mode can only specify argument list that were part of the input. There are only two combinatory possibilities for constructing new argument lists out of these, coming with the asymetry of append. No other operations on argument lists are definable. The number of flags to specify for the new lists is as restricted as it was at input. All the components of merge appear to stem from restricted sets. Consequently, the number of different combinations is bound to be finite, and so is the set Mods of merge modes.

The set Cats of categories in a particular grammar is not necessarily finite. Although every lexical item is assigned to a limited number of categories, Cats is closed under merge. Merges may create categories to which no lexical item is assigned. Each of these categories is of finite arity, but there is no limit imposed on this arity. Vijai-Shanker and Weir (1994??) have argued that limiting the arity is necessary for maintaining polynomial complexity. In part 3 this issue will be pursued further.

Notwithstanding the finiteness of the syntactic tools in Categorial List Grammar, the architecture sketched above allows for a subtle casting of natural language phenomena, as will be shown in part 2. By its being restricted the grammar imposes limits on the class of languages than can be described adequately by it. In this sense the grammar defines a nontrivial class of languages.
In this section several aspects of the grammar sketched above are discussed.

1.1.2.1.2.3. Appending argument lists: lists as stacks, lists as agenda

The only operations defined on argument lists are cancellation of an argument according to the modality that comes with it and appending. The latter operation creates a new list but leaves the lists that are feeding it intact. One could think of alternatives to this rigid form of list construction, like mixing or popping:

(21) mixing: $[a,b] \nabla [d,e] = [a,d,b,e]$
(22) popping: $[a,b] \lrcorner [d,e] = [d,a,b] \lrcorner [e] = [e,d,a,b]$
(23) appending: $[a,b] \oplus [d,e] = [a,b,d,e]$

Mixing is sometimes referred to as list merge; for obvious reasons here the term mixing is preferred. Lists are linearized and thus ordered structures. Specifying append as the operation to be performed while merging categories, implies conservation of these structures with respect to adjacency (being-next-to) and precedence. Mixing respects linearity but not adjacency. Popping respects adjacency, but partially reverses linearity. As a matter of fact, appending is the only operation that respects both the local characteristic of adjacency and the global characteristic of precedence.

Comparing mixing, popping and appending, some analogies are straightforward. Popping reflects the way stacks combine. A stack is loaded and emptied in a regular fashion, element by element, by simple repetition of the same maneuver. Mixing is a context sensitive operation. It requires keeping track of former moves and accounting for the internal structure of the lists involved. The analogy here is that of combining agendas, *i.e.* tasks that have to be performed in a certain order, feeding and bleeding each other. Appending, then, is in-between. It requires recursion, *i.e.* going back to a bottom case while suspending the shifts of individual elements. There is no bookkeeping of the internal structure of the lists involved. It is context free.

The conservation of adjacency and precedence in the local environment introduced by a category is the reason to consider append as the structural operation that merge lives on. Categories to which lexical

items are assigned, reflect partial knowledge of the combinatoric potency of that item. That knowledge concerns selection and subcategorization, but also linearization of the selected items. Moreover, it can be seen as constituting the domain that is governed by the phrase assigned to this category. This configuration is not accidental but defines the whereabouts in which the phrase can be interpreted. The merging of categories is the engine of phrase combinatorics in this grammar. As such it is bound to account for the discontinuities that natural language abounds of. But merging would be self-destructive if it would come with the mutation or covering of the configurations that make the phrases which it combines meaningful and interpretable. Grammar is supposed to make sense of the intrinsicalities of natural language. Appending the identifying domains of phrases, then, seems to be the less offensive option in merging categories.
For grammatical reasons, the definition of append has to be slightly adapted. See (88).

1.1.2.1.3. A SEQUENTIAL LOGIC

1.1.2.1.3.1. Deduction schemes

The grammar outlined here can be framed into a sequential, deductional format. Here is a model. Types are indicated by lower cast letters, lists of arguments by indexed L_i and R_j. Categories are marked by capitals roman capitals A, B, C. Possibly empty sequences of categories are marked by distinct capitals from the Greek alphabet.

(24)
(25) unary axiom: $C \rightarrow C$
(26) binary axiom: $A \otimes_i B \rightarrow C$ *for all $\otimes_i$ defined*
/* $a\backslash L_1/[b\text{^}i|R_1] \otimes_i b\backslash L_2/R_2 \rightarrow a\backslash L_1 \oplus_i L_2/R_1 \oplus_i R_2$ *for all $\otimes_i$ defined*
$b\backslash L_2/R_2 \otimes_j a\backslash[b\text{^}j|L_1]/R_1 \rightarrow a\backslash L_1 \oplus_i L_2/R_1 \oplus_i R_2$ */

(27) left rule

$$\frac{\Delta \rightarrow B \qquad A \otimes_i B \rightarrow C \qquad \Gamma', C, \Gamma'' \rightarrow T}{\Gamma', A, \Delta, \Gamma'' \rightarrow T}$$

(28) right rule:

$$\begin{array}{ccc} \Delta \rightarrow B & B \otimes_i A \rightarrow C & \Gamma' \; C \;\; \Gamma'' \rightarrow T \\ & \Gamma', \Delta, A, \Gamma'' \rightarrow T & \end{array}$$

Note that the right hand premise is shorter than the conclusion in that its antecedent contains at least one category less than the conclusion's antecedent. Note furthermore than in all binary axioms the arity of the consequent is exactly one less than the sum of the arities in the antecedent, which is necessitated by the definition of merge as involving cancellation. It follows that the arity of the right hand premise is smaller than the arity of the conclusion. Clearly, then, the decidability of this calculus depends on the decidability of the term $\Delta \rightarrow B$. We prove its decidability by induction on the length of Δ. $|\Delta| \leq |\Gamma' \; \Gamma''|$, by definition. If $|\Delta|=1$, the term instantiates identity or it is false. If $|\Delta| = 2$, its components satisfy, following the rule format, one of finitely many binary axioms, or it is false. If $|\Delta| > 2$, the rules apply to create a left hand premise $\Delta' \rightarrow B'$ such that $|\Delta'| < |\Delta|$. Given the remarks on the decidability of the middle and right hand premise in the rules, the derivability of each proposition $\Gamma \rightarrow t$ can be deduced in a finite number of steps.

1.1.2.1.3.2. A string model

The categories and the operations defined on them can be interpreted on a string model <L*, +>. L is the set of atomic phrases of a language and L* is the set of non-empty strings over that language such that L $\subseteq$ L* and for all a, $b \in$ L*, $a+b \in$ L*. The string operation + is taken to be associative and noncommutative. It does neither give rise to idempotency nor to persistency. As such, it is, as Morrill (1994) states, an operation on pieces of matter rather than an operation on pieces of knowledge. Another image that hits one's mind when looking for interpretations, is the flow of time. Languages essentially come in temporal extension. Phrases can be seen as decorated intervals. Although time may be dealt with under all kinds of logics, there is a standard perception of the flow of time as a noncommutative, *i.e.* irreversible, and nonpersistent, *i.e.* segmentable, process. The operation + is meant to reflect this analogy.
Consider furthermore the power set 2^{L^*}. A mapping $\Re$: C $\cup$ T$\rightarrow 2^{L^*}$ has to be established. In presenting categories for this purpose we will, for the moment, abstract from flags and modalities. The operator • indicates asymmetric associative concatenation of categories; ° interpretes this relation on categories.

(29) $\Re$(A) = $\Re$ (A\[]/[]) for all A $\in$ T
(30) $\Re([A_1,\ldots,A_i,\ldots,A_n]) = \Re(A_1\bullet\ldots\bullet A_i\bullet\ldots\bullet A_n) = \Re(A_1)^\circ\ldots^\circ\Re(A_i)^\circ\ldots^\circ\Re(A_n) = \{ a_1+\ldots+a_i+\ldots+a_n \mid a_j \in \Re(A_i), 1 \leq j \leq n \}$
(31) $\Re(A\otimes_m B) = \{ a+b \mid a\in\Re(A), b\in\Re(B)\}$, for all m
(32) $\Re$(A\x~L/y~R) = { a | ∀l∈$\Re$(reverseL), ∀r∈$\Re$(B) l+a+r∈$\Re$(A) } (the reverse of L is needed here as the order of the list of arguments to the left is the reverse of the order of strings associated with that list; flags of argument lists do not interfer with the denotation of categories.)

The syntax established here gives rise to a form of category merging that has been dubbed *mixed* or *disharmonic* composition. Its characteristic feature is that arguments from the secondary category not belonging to direction that is affected by the cancellation, are taken over by the new category. For example, if *Sla* below is non-empty, the indicated merge will involve this disharmonic composition.

(33) Prim\Plf~Pla/Prf~[Sec^j|Pra] $\otimes_j$ Sec\Slf~Sla/Srf~Sra $\rightarrow$Prim\Slf~Sla⊕Pla/1~Sra⊕Pra

The particular appending of *Pra* and *Sra*, on the other hand, is called *homogeneous* or *harmonic*. Here the arguments brought in by the secondary category, are of the direction affected by the cancellation. They are placed on top of those brought in at that side by the primary category. As a matter of fact, the syntax of merging listed first order categories under modalities induces several distinct patterns of discontinuity; these will be discussed in section XXX. The syntax is a syntax of discontinuity, rather than a syntax of concatenation. The basic combinatorics that is envisaged for strings, is chaining rather than appending. It is designed not so much to combine strings *w* and *z* into *wz* but to combine strings *wx* and *zy* into *wzxy* or *zwyx* in a controlled manner. Such forms of string merge are immanent to natural language: strings chain each other rather than glue together. They bubble up between each other's edges in a variety of modes. Nevertheless, we can take resort to + as the designated operation on strings, since no operation of the grammar violates the integrity of a deduced string. Merging prompts strings for peripheral association only. It does not involve any kind of extraction from or insertion in otherwise constructed strings. The discontinuous effects are solely caused by reference to the internal structure of the category that is associated with a string, *i.e.* to which a string is assigned. Once a string is derived by merge, it is maintained in the rest of the derivation. In this respect, the syntax is conservative and monotonous.
By consequence, the general strategy for constructing *ywzx* as stemming from *wx* and *yz* is mandatory. Suppose *wx* $\in \Re$(A) and *yz* $\in\Re$(B) for some A and B, but there is no C such that *wz*, *yw* or *zx* $\in \Re$(C). That is what it means that *ywzx* stems from *wx* and *yz*. The only derivation in the grammar depicted above runs as follows. Construct *w* and put the construction of string *x* on the agenda. Construct *z* and put the construction of string *y* on the agenda. Merge *w* and *z* and execute the agenda.

Now consider how under this regime a string *wyzx* must be derived from the same formants and with the same labour division between *w* and *x*. The string *wz* cannot be constructed, not even if for some C $wz \in \Re(C)$, since no rule of grammar could split up it up for *y* to be inserted. Therefore, the string *yz* has to be formed first. Next, *wyz* can be derived, while putting *x* on the agenda.

1.1.2.1.3.3. Disharmonic composition and categorial logic

As indicated above, CLG comes with disharmonic composition. In the present setting, disharmony results from appending a non-empty argument list of the secondary category in the passive direction, *i.e.* the direction that is not affected by cancellation - *Sla* in merge scheme (33). It has often been argued that categorial grammars involving disharmonic composition are beyond string models. Disharmony is rejected in, *e.g.*, Carpenter(1997), Morrill (1994) and Jacobson (1991). The latter states that disharmonic or mixed composition, though attractive for dealing with certain phenomena, cannot be function composition. If *f* sends *a* to *ba* and *g* sends *c* to *cd* and *cd* is in the domain of *f*, then $f^\circ g$ sends *c* to *bcd*, as Jacobson correctly claims for functions *f* and *g*. If the merge of categories A and B does not have this functional effect, their merge cannot be functional. The argument takes directionality of categories serious: it defines the functionality of categories in terms of linearization of strings. On the other hand, functions are not necessarily directed objects. The functional interpretation of categorial deduction under the Curry-Howard correspondence (cf. Van Benthem 1986) abstracts from directionality. As a matter of fact, this is the main reason why the morphism is not an isomorphism over substructural logic - intuitionistic logic with selective use of structural rules - and a particular fragment of the lambda-calculus. Merge can be seen as a complex of operations, some of which deal with linearization, while others take care of the logical aspects. The linearization operation - append, basically - does not contribute to functionality, although append itself is a homomorphism in $L^{L\times L}$, for L being the class of finite lists over a given domain. The grouping or bracketing operation that comes with merge, is as functional as can be.

Morril (1994: p. 231-232) refutes a syntax which applies selective linearization schemes: '… then the theory of syntax is not logical, in the sense of being the reflection of an interpretation of category formulas, but … a deductive system receiving definitions in terms of non-logical axioms and rules.' The point here is not only that one could choose, operationally, between redefining 'existing' operators and adding new ones, the latter being common practice in Multimodal Categorial Logic (Moortgat 1998). The crux is the concept of the relation between logic and grammar. In Morrill's understanding, logic starts out with irrefutable axioms and the grammatical combinatorics reflect this reasonable base.
The basic logic is intuistionistic conditionalization. It defines three operators (left divison, right divison and noncommatative multiplication) in a multiplicative fashion by residuation :

(34) $a \rightarrow c/b$ *iff* $a \cdot b \rightarrow c$ *iff* $b \rightarrow c\backslash a$

One could also chose for nondirected commutative operators, acoording to

(35) $a \rightarrow c|b$ *iff* $a \cdot b \rightarrow c$ *iff* $b \cdot a \rightarrow c$ *iff* $b \rightarrow c|a$

Other operators can be defined, also in duals, in terms of residuation. Famous by now are the one-place operators ◊ and □ (see Moortgat 1997, Morrill 1994):

(36) $\Diamond a \rightarrow b$ *iff* $a \rightarrow \Box b$

Their multiplicativity can be compared to the duals *n-root* and *n-power* defined on the positive reals: $\sqrt{a} = b$ *iff* $a = b^2$.

For deduction, division is interpreted as implication and multiplication as co-occurrence. The one-place operators add modal control, by closing and opening - basically, marking and unmarking - (sequences of) categories.
Algebraically it is worth noting that in the set of categories no unit element for multiplication is available. There is no type *t* such that $a{\cdot}t \rightarrow a$ or $a|t \rightarrow a$. In the set of strings the empty category may serve as the unit. But it is by no means certain that the empty string is part of the language itself.

From a logical point of view, an important aspect of this system is that it lacks negation, or its algebraic counterpart, complementation. Negation is the landmark of reasoning. It constitutes the minimal device one needs to generalize over situations or state-of-affairs. By consequence, natural language is not identified with reasoning as such, but depicted as an algebra forging some structure over a set of expressions. Linguistics, then, is about the components of this algebra.

The absence of complementation in the logical basis to the categorial analysis can be well defended under reference to the nature of language combinatorics. Language is a material system; it cannot be unwinded. It irreversebly consumes time and space, and it is hard to see what operation in language could be covered by algebraic complementation or logic negation. In the same vain, it is worth asking what in natural language makes conditionalization a suited vehicle to guide combinatorial operations. The answer, as it seems, hides in the previous recognition that language is a material, resources consuming system. To say that a certain element requires another or is conditionalized by another, is saying that somewhere in the linguistics space (*e.g.* a sentence) that other element must be available. *Somewhere*, however, is far too weak. It never means *anywhere*. Generally, the requirement is much more specific: the other element must be available in a certain well defined subspace of the linguistic space. This subspace is identified relatively, with respect to the position of the requiring element, and in linear terms: to the right or to the left of that element. Relative linear occurrence is the language instance of logic conditionalization. In natural language, linearity is conditionality. It is by no means accidental that direction is part of the logical basis for natural language analysis. Directionality is the name of the language game. And it is from directional requirements that we derive hierarchy, not the other way around. Such is the message of Kayne (1994). We are still far from understanding exactly how linearity induces subordination and semantic dependency, but linearization is indispensable in the grammatical base.

This being the case, we have at least two datastructures for encoding the linearization requirements of phrases, *i.e.* their spaced conditionalizations.
First, there are the types exploited in Lambek (1958) and much subsequential related work. Combinatoricaly, they can be seen as onions. Their internal structure might be complicated, but only their outer peel determines the combinatorial potential in a given state. The related combinatorics is naturally context free, as the internal structure of the consequent of the primary category in the directed *modus ponens* that drives the system, is not considered. The computations that come with it relate to a tuple <body, outer peel>. As outer peel must be seen the specified argument *with* its direction (see Steedman 1990 for an invariant of this nature).
Second, there are the 'flat' categories as derived in Buszkowski (1988). Though the author not necessarily will see them this way, they introduce the full directed internal structure of the *premisse major* as an entity in the *modus ponens*. Combinatorics defined on these datastructures are context sensitive (though not necessarily in the strong sense) because the internal structure of the premise is specified and possibly taken into consideration. The computations that go with it relate to triples <head, leftward structure, rightward structure>. These characteristics of the flat, peeled categorial data structures also extend to the very first proposals in the Combinatory Categorial Grammar style, in Ades & Steedman (1980).
These two types of categorial information structures induce different calculi. The differences follow from the data structures itself. The flat types allow for a fine grained resource management in terms of linearization and subspaces. The onion types induce calculi that exploit the bare conditionalization, in the

form of a conditional hierarchy, adding additional operations for resource management like linear structering.

- the absence of negation: language elements cannot be unwinded, language is resource consuming
- linearization is the conditionalization of language
- datastructures: onions(Lambek) vs pancakes (Buszkowski): translatability
- stressing conditionalization vs stressing linearization
- compuatation considers pairs <body, outer peel> vs compuatation considers triples
- context free(not considering the internal structure of the body of the primary category vs. considering the internal structure of the body of the internal category)
- allowing for fundamental calculus with added operations or modalities vs allowing fine grained resource management

The strategy for characterizing natural languages as an algebra is to choose some well defined starting point and see which additional power is needed to deal with the intrinsicalities of the language at hand. That starting point could be LP, the categorial system based on the commutative residuation (35) and explored by Van Benthem (1991); it amounts to the Lambek calculus under abstraction of directionality. This system, though, is to weak. It defines no (syntax of any) natural language at all, since it induces a permutation closure over the set of accepted strings accepted. No natural language is known to be that free, not even Walpiri, Hungarian or Latin. Thus it seems reasonable to upgrade one level, and exploit P, the system defined by residuation (34), i.e. the Lambek calculus. Doing so, we take directionality and non-commutativity as basic, in the same vain as the interpretation of division as conditionalization is basic. On the other hand, directionality and non-commutativity can be added to LP in terms of modalized duals. Here is an example, expressing the functionality of (34) in (35):

(37) $a \rightarrow c|_i b$ *iff* $a \; {}_{i \cdot j} \; b \rightarrow c$ *iff* $b \rightarrow c|_j \; a$

Here we can see that directionality and non-commutativity can be added to LP, just like any other regime of structural management could be. The expressivity of such a system is more than sufficient to cover natural languages, as it seems. It leaves room for the embedding of complex analyses of natural languages, as shown in Vermaat (1999) and Stabler & Retoré (1999). In this approach, as it seems, every computable structure can be accommodated. It is Turing complete. Of course, for every particular phenomenon or analysis, it is highly interesting and revealing to scrutinize which embeddings are possible. In fact, the multimodal approach exemplified in (37) seems to come with the categorial alternative to the Chomsky hierarchy called for in Van Benthem (1991).
CLG is considerably less expressive than multimodal categorial grammar. It imposes heavy restrictions on the set of derivable sequents, and moreover, for any finite lexical assignment, on the class of accepted strings. CLG, thus, defines a subset of the class of partial recursive languages (see Cremers 1999). This is achieved by build-in discontinuity.

Morrill's stand considers grammar as applied logic, as living on logic grounds. In this view, adding axioms or adjusting operators just to comfort language analysis is needlessly weakening the grammar-logic connection; it amounts to redefining mathematics in order to get grip of nature. In another perspective, however, logic is defining (some of) the instruments for grammatical construal. Grammar is not necessarily applied logic, but may involve the application of logical means for natural language analysis. This perspective finds justification in the present state of ignorance about the embedding of language in the cognitive and intellectual resources of our species. As far as I can see, we have at this moment conclusive evidence neither for language as an independent faculty nor for language as an epiphenomenon of emerged cognitive capacities nor for anything in between. This should not withhold us from pursuing one or other hypothesis. In any case, one should adhere to making explicit in grammar all those processes that one observes in scrutinizing natural language. Logic is extremely useful in this explication. CLG is an effort to localize the interference of selection and linearization in natural languages. Take, for example, the wrap that typical auxilaries in Dutch induce. Auxilaries, except when fronted to second position, are typically head adjuncts: they select a verbal complement of some sort to their right, but will typically wrap this complement around itself in such a way that the complement's head is its right neighbour:

(38)	auxiliary	*wordt* ('is', passive aux)	x/vppas
(39)	vppas	*in de verf gezet* ('painted')	verb: *gezet*
(40)	string	in de verf wordt gezet	x

Alternatively, they select a complement to their left. But in that case everything occurring to the right of the complement's head has to occur, after merge, to the right of the passive auxiliary:

(41)	auxialiary	*wordt*	x\vppas
(42)	vppas	*met geweld gedwongen te vertrekken*	('violently forced to leave')
(43)	string	met geweld gedwongen wordt te vertrekken	x

Neither for the rightward nor for the leftward selection there are string ordering alternatives. Every other ordering essentially affects interpretation or endangers interpretability. The following ordering variations on leftward selection show this.

(44) * met geweld gedwongen te wordt vertrekken
(45) met geweld gedwongen te vertrekken wordt
can only mean something like changes into 'violently forced to leave'

But then the following statement appears: if the direction of the complement selection is part of the auxiliary's lexical definition, the way of wrapping that complement is part of that definition too. That is what CLG specifies. It takes discontinuity serious, in defining natural language grammar as the computation of dicontinuity. In this vain, nonadjacency of strings that are to be interpreted in relation to each other is a fundamental fact of language. Language is linear. It is a time consuming process, and the order of events in this process is crucial. Disharmonious composition is one of the instruments to deal with this ordering. It is not favoured or disfavoured to other merge formats. Logic, then, is just exploited to keep the analyses tractable.

1.1.2.1.3.4. Merge and Move

The presentation of the syntax till now leaves no room for a fundamental distinction between move and merge operations, as introduced in Chomsky (1995). Stabler (1996) essentially reduces the distinction to a matter of arity. Move is operating on a category to produce another. Merge operates on two categories to produce another. In both operations checking features is the engine of the process. Cormack and Smith (1999) hypothesize that interpretation and merge of a constituent take place at the same level of derivation. There is no need for move, then. CLG incorporates the view that merging implies moving, as lexical material is anchored in positions where it may or must split what otherwise would heave been fine chains. The famed case at stake here is dative insertion. It is generally acknowledged that a verb and its direct object are tightly related. In many languages, however, a second object - the indirect one - must or may interfere between the verb and the direct object. This object too is licensed by the verb, but it seems to be looser related to it, syntactically and semantically. There is no particular relation between the direct and the indirect object. But it is also not the case that the complex of verb and neighbouring indirect object represent a more complete syntactic or semantic frame to the direct object. I don't know of any cases where verb and indirect object bring in interpretations that are not available to the verb without indirect object. Since the indirect object is still part of the verbal complex, the mere merge of verb and indirect object induces relative movement of the direct object to some distance from its very licenser, the verb. In the same vein, one might look at the relation between prenominal adjectives and determiners. It is quite clear that there several very severe restrictions operative between a determiner and the noun, ranging from morphological shape to the quantificational nature of the resulting constituent. Adjectives have no or hardly any relation to the determiner, and some, but not very lexical (computable) semantic cross-links to the noun. Still, adjectives must or may occur at that side of the noun where determiners are bound to draw. The complex formation of adjective and noun, then, puts the determiner at a distance from its closest comrade. In may respects, therefore, merge inevitably induces move.

1.1.2.1.3.5. Soundness and completeness

A sequent of the form $\Gamma \to B$ is valid if the set of phrases that is associated with Γ in included in the set of phrases associated assigned to B, *i.e.* if $\Re(\Gamma) \subseteq \Re(B)$. Here is a proof that the sequent calculus presented above, is sound and, in a certain sense, complete with respect to the string model.

- Soundness

As for soundness, *i.e.* the proposition: what can be derived is valid, consider the following. The unary axiom is trivially valid. The binary axioms are valid by definition. We can therefore concentrate on the left one of the two only productive derivation steps

(46)
$$\frac{\Delta \to B \qquad A \otimes_i B \to C \qquad \Gamma', C, \Gamma'' \to T}{\Gamma, A, \Delta, \Gamma'' \to T}$$

Suppose $d \in \Re(\Delta)$ and $a \in \Re(A)$. By induction, $d \in \Re(B)$. By definition, $a+d \in \Re(C)$ and $\Re(A \bullet \Delta) \subseteq$. So for $g' \in \Re(\Gamma')$ and $g'' \in \Re(\Gamma'')$, $g'+a+d+g'' \in \Re(T)$ and thus $\Re(\Gamma \bullet A \bullet \Delta \bullet \Gamma'') \subseteq \Re(\Gamma' \bullet C \bullet \Gamma'') \subseteq \Re(T)$.

- Completeness

As for completeness, *i.e.* the proposition: what is valid can be derived, we have to prove that $\Re(\Delta) \subseteq \Re(T)$ implies derivability of the sequent $\Delta \to T$. In its full strenght, this probably can not be proved for the present system, as it lacks hypothetical reasoning. What can be proved, however, is a considerably weaker statement that amounts to the following proposition. There is always an assignment of phrases to categories such that if $d \in \Re(\Delta)$, $d \in \Re(T)$, then also $d \in \Re(\Gamma)$, $|\Delta| = |\Gamma|$ and $\Gamma \to T$. In other terms: for every Δ such that $\Re(\Delta) \subseteq \Re(T)$, there is a Γ such that $|\Delta| = |\Gamma|$, $\Re(\Delta) \subseteq \Re(\Gamma)$ and $\Gamma \to T$. This means that the calculus is complete *salve* assignments of phrases to categories. The proves gives a construction of Γ as a string of categories $\Delta[X_i \leftarrow Y_i]$ which is like Δ except for a finite number of substitutions of a category X_i by Y_i. The number of substitutions has an upper limit $|\Delta|$ and consequently, $|\Delta[X_i \leftarrow Y_i]| = |\Delta|$.

- Help lemmas for completeness *salve* assignments

To prove completeness *salve* assignment, we take resort to the following lemma; here *L-LL* denotes the remnant of L after taking out what L and LL have in common.

(47) binary derivability lemma
for all Γ, $|\Gamma| \geq 2$, $\Gamma \to$ a\L/R *iff*
$\Gamma = \Delta' \Delta''$ *and either*
$\Delta' \to$ a\L_1/[b^i|R_1] , $\Delta'' \to$ b\L-L_1/R-R_1 and

$$a\backslash L_1/[b\hat{\ }i|R_1] \otimes_i b\backslash L\text{-}L_1/R\text{-}R_1 \rightarrow a\backslash L/R$$

or

$$\Delta' \rightarrow b\backslash L\text{-}L_1/R\text{-}R_1,\ \Delta'' \rightarrow a\backslash[b\hat{\ }i|L_1]/R_1 \text{ and}$$
$$b\backslash L\text{-}L_1/R\text{-}R_1 \otimes_i a\backslash[b\hat{\ }i|L_1]/R_1 \rightarrow a\backslash L/R$$

This lemma is supposed to play the same role in the present completeness proof as the *canonicial lemma* of Buszkowski (1982) in the completeness proof of the product free Lambek calculus.
From right to left the lemma is evident, as it reflects the deductive aspect of the calculus. From left to right the lemma is not trivial. It implies binary branching of the deductive proces. The crucial deduction step above substitutes a category *C* by a string A,Δ. If $\Delta \rightarrow B$ is derivable, it is an axiom or it is derived by the same rule. In that case, Δ can be represented as Δ'', B' or B', Δ' for some category B' with the same head as *B*. This can be repeated till we end up with single categories only. Thus the structure 'below' *C* can be seen as a binary branching construal with a category on one branch and a , again branching, string of categories at the other. 'Above' *C* we may safely assume that C itself will be a right hand of left hand partner of some peripheral substring of either Γ' or Γ''. This duet, then, is substituted, in deduction, by come category C'. Consequently, *C* itself will be the product of a binary process. Since every reduction decreases the number of categories, the structure converges, and must be binary branching. But then, at the top of that structure, there are two categories covering the whole string of categories that is deduced. The format of the categories involved in the lemma, follows from the definition of merge. Note that by the monotony property of the grammar the arity of the consequent puts an upper limit on the arity of each of the categories in the antecedent.

We need furthermore the notion of *compatibility* between categories. If $A \otimes_i B \rightarrow C$, both A and B are compatible to C. If *A* is compatible to *C*, then, by binary derivability, for some *B*, $A \otimes_i B \rightarrow C$. One can easily compute that compatibility of *A* to *C* amounts to *A*'s argument lists being embeddable in *C*'s. It is noteworthy that the number of categories X compatible to a given Y is finite: Y's argument lists are finite and the set of types is finite. In fact, the number is linear in the arity of Y.

Finally, it must be shown that for every category that could be the product of reduction by some merge, such a merge can be found. A category is *reduced* if exactly one of its argument list is flagged 1. Recall that this flag indicates that one of the argument list from which it was appended, contained an argument which was cancelled by merge. If none of the list flags of a category is 1, the category might be a reduction nevertheless; this would, however, depend on the availability of particular instances of merge (see XXX for an example). Furthermore I assume that there is no interesting difference in denotation between two categories that only differ as to the flagging of their arguments. Flagging is control, not semantics. This was expressed in statement (32)of the definition of $\mathfrak{R}$.
Under this *proviso*, the following has to be shown:

(48) reducibility
for every reduced *T* there are categories *A* and *B* and a merge mode *i* such that
$A \otimes_i B \rightarrow T$

Since categories are constructable *ad lib*, the presence of suited merge modes is the bottleneck here. We may safely assume that every grammar defines at least one merge mode. Let $\otimes_i$ be this merge mode and suppose, without loss of generality, that it induces a right hand side cancellation; let *T*'s reducability assent to this directional feature. *T* will, then, have the format $a\backslash fl_{lt}{\sim}L/1{\sim}R$. Thus it must be proved that there are argument lists L_1, L_2, R_1 and R_2 such that:

(49) $a\backslash fl_{la}{\sim}L1/fl_{ra}{\sim}[b\hat{\ }i|R_1] \otimes_i b\backslash fl_{lb}{\sim}L2/fl_{rb}{\sim}R2 \rightarrow a\backslash fl_{lt}{\sim}L/1{\sim}R$

The merge mode *i* specifies some appends $L_1 \oplus_{il} L_2$ and $R_1 \oplus_{ir} R_2$ with appropriate flagging as ouput conditions and some restrictions *inp*(L_j) and *inp(Rj)* as input conditions on the left and right argument list of the two antecedent categories, inclusing their flags. Now suppose that fl_{la}~*L1* agrees with *inp*(L_j) and that fl_{ra} ~[*b^i*|R_1] agrees with *inp*(R_j); of course we can always construct the first category to accord with the input requirements to *i*. So it remains to show that given *T*, ⊗i and an appropriate compatible *A*, *B* can be found, to wit, be constructed. Recall that the appendings that come with the merges, are defined conservatively. They are restricted to asymmetric linear gluing of lists. Consequently, L_1 is constructed is such a way that it is a sublist of *L*; as a matter of fact, it is a (possibly empty) prefix or a (possibly empty) suffix of *L*. Which of these options holds, is defined by $\oplus_{il}$ as part of the definition of $\otimes_i$. But then, L_2 is the result of subtracting *L-L1*. This subtraction is uniquely defined given *L* and L_1. The same reasoning carries over to R_1, R_2 and *R*. So, L_2 and R_2 are uniquely defined by $\otimes_i$ and the other argument lists. As for the flags of the argument lists of the antecedent categories: since they do not reflect in any respect the present state of these lists, they can be chosen freely in accordance with $\otimes_i$. So, *B* can be constructed. This proves reducability.

❑ Completeness *salve* assignments

Given binary derivability, completeness *salve* assignments amounts to the claim that $\Re(\Delta) \subseteq \Re(T)$ implies derivability of the sequent $\Delta', \Delta'' \rightarrow T$ for some bipartition Δ', Δ'' of Δ and reduced *T*. Recall that $\Re(T) = \Re(T')$ if $T = T'$ *modulo* flagging of argument lists. Now suppose d ∈ $\Re(\Delta)$, d = d'+ d'', d' ∈ $\Re(\Delta')$, and d'' ∈ $\Re(\Delta'')$. Then, if $\Delta' \rightarrow A$ and $\Delta'' \rightarrow B$ for some *A* en *B* such that $A \otimes_i B \rightarrow T$, $\Delta', \Delta'' \rightarrow T$ and, by soundness, d'+d'' ∈ $\Re(T)$. So it must be proved that such *A* and B exist. This proof is by induction on the length of Δ' and Δ''.

(50) If $|\Delta'| = 1$, we have identity and for some *A*, $\Delta' \rightarrow A$.

(51) Suppose A is compatible with T. By reducibility, there is a *B* such that $A \otimes_i B \rightarrow T$. As for Δ'', the proof now requires $\Delta'' [X_i \leftarrow Y_i] \rightarrow B$ to be derivable.

(52) Otherwise, substitute for A a category A' that is compatible with T. Assign d' to that category, *i.e.* assure that $\Re(A) \subseteq \Re(A')$. Clearly $\Delta[A \leftarrow A'] \rightarrow A'$. Proceed with $\Delta'' \rightarrow B$ such that $A \otimes_i B \rightarrow T$. By reducability, there is a *B* to decide on.

(53) If $|\Delta''| = 1$ and $\Delta'' = C$ and $C \rightarrow B$, we are done. Either $\Delta \rightarrow T$, as in case (51), or $\Delta[A \leftarrow A'] \rightarrow T$.

(54) But suppose that not $C \rightarrow B$.

(note: here hypothetical reasoning would do miracle, as it would assign to B also T\A. But we don't have hypothetical reasoning or introduction).

We assign every string in $\Re(C)$to $\Re(B)$. Then, $\Re(C) \subseteq \Re(B)$ and in particular, d'' ∈ $\Re(B)$. It is evident that substituting *C* for *B* in Δ '' amounts to identity, and assures derivability of $\Delta[C \leftarrow B] \rightarrow T$, *i.e.* reducability of the string Δ with the rightmost occurrence of *C* substituted by *B* to *T*. Moreover, d ∈ $\Re(\Delta[C \leftarrow B])$ and d ∈ $\Re(T)$.

(55) Now suppose $|\Delta''| = 2$. Again, by binary derivability, $\Delta'' \rightarrow B$ if C' $C'' \rightarrow B$ for some C', C''. We take the same cycle (50)-(54) as above and show that either $\Delta'' \rightarrow B$ or $\Delta''[C' \leftarrow C''''] \rightarrow B$, or $\Delta''[C'' \leftarrow C''''] \rightarrow B$, or $\Delta''[C' \leftarrow C''', C'' \leftarrow C''''] \rightarrow B$. Moreover d'' ∈ $\Re(\Delta''[\ldots])$ and d'' ∈ $\Re(B)$. Consequently, d'+d'' ∈ $\Re(A \otimes_i B) \subseteq \Re(T)$.

(56) If $|\Delta''| > 2$, the reasoning (51)-(55)applies to prove the derivability of some sequent $\Delta''' \rightarrow B$, where Δ''' is Δ'' except for a number of substitutions $X \leftarrow Y$ linearily bounded by $|\Delta''|$, such that d'' ∈ $\Re(\Delta''')$, $\Delta', \Delta''' \rightarrow T$ and d ∈ $\Re(\Delta' \Delta''')$.

(57) Now suppose that *A* is not compatible with *T*. Then substitute A by some A' that is compatible to *T* and assure that $\Re(A) \subseteq \Re(A')$. Then go for B, as above. It is inevitable that $\Delta[A \leftarrow A'] \rightarrow T$ or $\Delta[A \leftarrow A'|Other] \rightarrow T$, where Other is any combination of substitutions induced by inspection of Δ''.

(58) Suppose |Δ'| > 1. Then following the track (55)-(56) above for Δ'', create a derivable sequent *Δ''''* *→ A* such that A is compatible with T, d' ∈ 𝔑(Δ'''') and Δ'''' is like Δ' except for a number of substitutions *X←Y* that is linearly upward bounded by |Δ'|. For that *A*, there must be a *B* such that *Δ''' → B* where Δ''' comes from Δ'' as described above. Again, d'+d'' ∈ 𝔑(Δ'''', Δ'''), *Δ'''',Δ''' →* *T* and Δ is like Δ'''',Δ''' except for a finite number of substitutions *X←Y* linearly bounded by |Δ|. Moreover, |Δ| = |Δ'''', Δ'''|.
This ends the proof of completeness *salve* assignments for Categorial List Grammar.

So, for an string *Δ* of power *n*, for which 𝔑(Δ) ⊆ 𝔑(T), one can find in at most *n* substitution steps, each of which is decidable, a string $\Gamma = \Delta[X_i \leftarrow Y_i]$, $0 \leq i \leq n$, such that *Γ → T.*

1.1.2.1.4. THE FUNDAMENTAL ASYMMETRY OF MERGE

In section 1.1.2.1.2.3 it was attested that append treats argument list, originating from two antecedent categories in a merge, necessarily asymmetric. At each side, all the arguments of one category will be on top of the arguments of the other category. This relative order correlates with relative distance of argument strings to the string of the category: the types on top of the argument list induce strings that will be closer to the string induced by the category's head than the strings induced by lower types. Since append is the canonical operation on argument list, this asymmetry is built in CLG.
Induction of a string s by an argument A of category C means that s ∈ 𝔑(A) or s ∈ 𝔑(Δ') such that for some Δ'', 𝔑(Δ'•Δ'') ⊆ 𝔑(A).
Furthermore, the linearization of the strings in the two antecedent categories of each merge brings in another intrinsic form of asymmetry. It implies that at least at one hand side the strings induced by the arguments of an antecedent category will be separated from the string induced by the category's head. Here is a scheme of this pattern for a certain instance of a defined merge mode.

(59) ($\otimes_i$/)
Prim\Plf~Pla/Prf~[Sec^i|Pra] $\otimes_{i\_}$ Sec\Slf~Sla/Srf~Sra →
Prim\Slf~(Sla⊕Pla)/1~(Pra⊕Sra)

(60) s\0~[np^j]/ 1~[vp^i, pp^k] $\otimes_i$ vp\ 0~[ap^l] / 0~[vp^m] →
(61) s\0~[ap^l, np^j]/1~[pp^k, vp^m]
(62) prim ∈ 𝔑(s\0~[np^j]/ 1~[vp^i, pp^k]), sec ∈ 𝔑(vp\0~[ap^l]/0~[vp^m]), vp ∈ 𝔑(vp^m), np ∈ 𝔑(pp^k), np ∈ 𝔑(np^j), ap ∈ 𝔑(ap^l)
(63) np + ap + prim + sec + pp + vp

Here, (63) represents a string that under assignment (62) may be the result of a derivation involving instantiation 0 of merge mode (59) . The string *prim* is of the category that licensed *pp* but is seperated from it by *sec*. Similarly, *sec* is of the category that brought in *ap* but is not adjacent to it in (63) ; yet, the type of *ap* was the top of the left argument list of the secondary antecedent category in merge 0. In this case, *sec* is not adjacent to any string induced by arguments of the category it belongs to. The string that is in the primary category, however, will by definition of merge be next to at least one string induced by the primary category's argument, to wit, the string induced by the cancelled argument and the secondary category. This asymmetric aspect of merge gives rise to a principled difference in status between strings of the primary category and strings of the secondary category. I hesitate to identify this difference with headness, as made relevant to categorial grammar by Pollard (1984), Hoeksema and Janda (1988) and Jacobson (1991), among others. A head is the predominant string in a certain substring. Its category defines the structure immediately around it. From the asymmetry observed here, we can only derive the following property of string connectedness for strings in primary categories of a merge.

(64) A string s $\in \Re(C)$ is connected in w+s+v if both w and v are induced by arguments in C, or w is induced by an argument in C and v is induced by an argument of the category w belongs to, or the other way around

Merge imposes string connectedness on the string in its primary category. The string in the secondary category lacks this property, as one of its neighbors, to wit *prim*, is not induced, directly or indirectly, by one of this category's arguments. In precisely this sense, the primary category defines a string beyond the string in it.
String connectedness has dual. If a string w+p+s+v is 'felt' to be defined by p then there is a category C such that p $\in \Re(C)$ and C connects p in w+p+s+v. It is not necessarily the case, however, that w+p++s+v is a constituent. There does not need to be also a sequence of categories Δ such that w+p+s+v $\in \Re(\Delta)$, that C is in Δ, that $\Delta \rightarrow C'$ for some C' (and that C introduces the head type of C'). If a 'head' defines a string, any of its substrings containing the 'head'-string is defined by it.
The basic asymmetry that comes with the *head-nonhead* distinction, is encapsulated in the very definition of the axioms of CLG.
Summarizing, merge imposes asymmetry in the relation between the antecedent categories in each of the following senses:

- precisely one category has one of its arguments cancelled, *i.e.* one category is primary, the other secondary
- the linear ordering of the categories in the merge induces the linear ordering of the strings
- append on argument lists is asymmetric, *i.e.* at each side the arguments of one category are on top of those of the other category
- the strings associated with the primary category are connected in the string defined by the primary category

Asymmetry, *i.e.* irreversibility of the (dependency) relation between elements, in natural language is the resultant of linearization. In CLG, linearization is an intrinsic component of the syntactic engine. It has been argued that in the grammar of natural language precedence and dominance relations could or should be separated. This has been a major topic in the development of GPSG and HPSG. It was suggested for categorial grammar by Flynn (1983). Flynn stressed that constituency relations might be at an other level of universality than linear restrictions. The topic reoccurs in the categorial mainstream in the format of specialized permutation modalities, meant to rearrange word order in a controlled way but indepedent of the combinatory, reductionist engine; see Moortgat (1998) for a detailed overview, and Hepple (1990) for the original approach. The modular view on precedence and dominance, word order and constituency, permutation and composition or any other duality that captures the two-dimensionality of natural language, has been challenged by Kayne (1994). Unlike the structure of the universe (Icke 1996), Kayne holds language to be essentially antisymmetric (antisymmetry is just a strenghtening of asymmetry) because (asymmetric) linearization *entails* dependency. In his view, a certain linearization is not an accident of a particular language, but the prerequisite to interpretability and its reflection. The present system is hardly reminiscent to Kayne's concept of grammar. But CLG respects the interdepency of word order and constituency by exploiting only one syntactic operation that accounts both for word order and for hierarchical, interpretative dependencies. The asymmetry which CLG imposes on the linear dimension and the ordering of strings, comes with an irreversible inequality of the primary and the secondary category in a merge.
The asymmetricality of CLG is, of course, not an argument in favor of linearity as the anchoring dimension of natural language grammar. It represents, though, a choice in this respect.

directionalty as anchoring dimension
the grammar of discontinuity
the necessity of flagging: lexical vs derived categories

subtility
detailed considerations of the syntax

1.1.2.2. Rules of Syntax

1.1.2.2.1. General Format

In section 1.1.2.1.2 the general properties of the standard two place operation of merge were introduced. It was noted that every merge modality specifies a format for the two categories that go into the merge and a format for the category that emerges from them. Several aspects of the operation are general. There is always exactly one type to be cancelled, in a designated position at the top of one argument stack. The stacks are adressed as lists. Cancellation requires typological identity of that argument and the head of the other stack. The head of the output category is invariably the head of the primary input category. The (remainings of) the argument lists are appended, pair- and directionwise. Argument lists are marked for affectedness, *i.e.* for having or not having undergone cancellation of one or more member types. Arguments are marked with one merging mode, indicating a specific mask on the input and the output of the merge operation that is to cancel that argument.

Furthermore, the nature of the specifications that can be made in a merge modality is fixed. They come from a limited set. Argument lists of the input may be required to be empty or not-empty; non-emptiness means that the stack contains at least or exactly one argument. specified or not - the latter difference will be ignored in the calculations. Moreover, list are marked affected or not-affected, initially unaffected by assignment and subsequently affected or unaffected by computation or specification. Specifications on argument lists are not necessary, though. The output category is specified in terms of the input components. The way of appending the input lists is determined. So is the affectedness marking on the output lists. Here is an overview of the possible specifications on input and output for a rightward cancelling modality *i*.

(65) Prim \ LPF~LP / RPF~[Sec^i| RP] $\otimes_i$
Sec\ LSF~LS/RSF~RS $\rightarrow$ Prim\ LF~LA / RF~RA

specifications of LPF, RPF, LSF, RSF: *affected*, *unaffected* or none
specifications of LP, RP, LS, RS: *empty*, *nonempty* or none
specifications of LF: *affected* or *unaffected*, depending on the values of LPF and LSF and the values of LP and LS
specifications of RF: *affected* (normally; exceptionally: *unaffected*)
specifications of LA: $LP \oplus LS$ or $LS \oplus LP$
specifications of RA: $RP \oplus RS$ or $RS \oplus RP$

Now we can compute the total number of different modalities that can be specified within these limits. There are twelve items that may be specified: six argument lists and six flags. For each of the input arguments lists are three options available; the options are formally independent. For each of the input flags also three options are available; the choices are, again, formally independent of each other. For each of the four components of the output category only two options are available, since they must be determined. These options are also independent, with the possible exception of the value of RF. Any mix of the options will define a modality. Consequently, within this format at most $3^8.2^4$ = 104976 different modalities can co-exist. Materially, however, this limit is too high. Every requirement as to the emptiness of an input argument list fixes the value (for append) of the output list in that direction, since $[\ ] \oplus L = L \oplus [\ ] = L$ for any L. Thus, the values for LA and RA are not independent of the values for LP and LS and for RP and RS, respectively. Practically, the limit then is $(3^2\text{-}2^2).3^4.2^2(\ 2^4 + (3^2\text{-}2^2))$ = 34020 different modalities for rightward merge.
Moreover, the flags of the output argument lists are completely determined by the values of the flags and lists of the input categories. They are therefore not so much part of a specification but computed at each merge, with the exception of RF's value being *not-affected*. This brings the number of modalities down to half the computed number, *i.e.* 17510 and almost to one quarter (say, 9000) if we take the exception to be exceptionally. Moreover, it is likely that we could establish various kinds of clusters of specifications, which might exclude or induce each other. For example, it is unlikely that a certain argument list is required to be both empty *and* affected or unaffected. Affectedness indicates that the phrase bearing such an argument list, is not lexical but composed. Unaffectedness has the opposite flavor. How likely is it that we discover a merge process that imposes conditions both on the internal structure of a category's component and on its 'flattened history'?

A categorial list grammar will be any set of modalities, defined on a given set of types. This view raises all kind of questions as to the consistency of these sets, as well as with respect to the practical and theoretical equivalence between classes of modalities, and the decidability and expressability of certain properties. Most of these questions cannot be addressed here.
In section XXX on patterns of discontinuity we will reconsider the variety of merge modalities. But clearly the amount of possible modalities makes no grammar constructed from them trivial. Each selection of modalities, even if the selection is very small, as will be the normal case, represents a very specific grammar.

1.1.2.2.2. A GRAMMAR OF DUTCH

All the modalities used in the Delilah grammar of Dutch, enjoy some kind of input specification. The single mode that does not specify any input component, is not used. It is unlikely that any reasonable grammar of any language will employ that mode. It amounts to the absolute insensitivity of an element for the structure of its environment. Only extra-sentential elements like interjections of sighing qualify for this form of context freeness.
The output category to the merge is always fully specified. The flags of the output argument list, appended from a pair of input lists, are computed from the flags of the input lists that are appended, and from the lists' status. The flag in the active direction of the merge will almost invariably get the value *affected*, abbreviated *a~* or *w~*. The affectedness value *w~* is assigned if the cancelled argument required application of the *wh*-modality. Most other cancellations lead to the affectedness value *a~*. This will be explained below. The other value is marked *u~*, for *unaffected*. The other output flag, the one for the argument list in the passive direction, is computed deterministically according to scheme (66).

(66) output argument list flag table

	input 1	*input 2*	*output flag*
(66.1)	w~_	a~[_|_]	w~
(66.2)	u~_	w~[_|_]	w~
(66.3)	u~_	a~[_|_]	a~
(66.4)	_~[ ]	_~[ ]	u~
(66.5)	u~_	_~[ ]	u~
(66.6)	u~_	u~_	u~
(66.7)	a~[_|_]	a~_	a~

The situation that two input lists are flagged *w~* does not occur, by the way the *w~* flag is embedded. The grammar assigns the flag *w~* to empty lists only, but the flag can be assigned to nonempty lists according to (66.1). Derivations involving the computations (66.1) and (66.2) are bound to be pathological and dead-ending, however. The most important aspect of the computation is the restoration of unaffectedness by (66.4) and (66.5). The moral is that whenever an empty list is appended, the nature of its emptiness - emptied or lexical - gets out of sight. This embodies a sort of locality. If a constituent has completed its agenda at one side, the next merge renders its completion invisible or irrelevant for further merges; its history and its structure is wrapped up in the new structure. Note furthermore that the output list can be marked *affected* only if at least one of the input lists is.

Below all the rules of the Delilah grammar are presented, explained and exemplified. The modalities are indicated by short memo-subscripts on the merge-operator and as extensions to arguments. The rules are named after the modalities. The flag of the output list in the passive direction is supposed to be computed according to the scheme above, and is not specified in the rule, but invariantly dubbed *Fl*.

1.1.2.2.2.1.

1.1.2.2.2.2. Rightward rules

❑ /^isl

This mode requires the secondary category to be accomplished. It does not allow of any part of a secondary agenda to be transferred to the output.

(67) Prim \ _~LP / _~[Sec^*isl*|RP] $\otimes_{\mathrm{isl}}$ Sec \ _~[ ] / _~[ ] $\rightarrow$ Prim \ Fl~LP / a~RP

Because of the emptiness of the secondary argument lists, the appends are trivial. The standard application of this merge is the comsumption of an noun phrase. Noun phrases are closed domains in Dutch. They don't give rise to any discontinuity of their components. All noun phrases in argument lists select the *^isl* cancelling mode.
Here is an example of *^isl* :

(68) *wil* 'wants' *de man* 'the man'
q\u~[ ]/u~[np^*isl*, vp^*x*] $\otimes_{\mathrm{isl}}$ np\ u~[ ]/a~[ ] $\rightarrow$ q\u~[ ]/ a~[vp^*x*]
wil de man 'wants the man' : does the man want …

Other candidates for cancelling under *^isl* are all those constituents that have been identified as absolute islands, like (embedded) questions. One of the standard categories for a verb selecting an embedded question must be **vp\u~[ ]/u~[q^*isl*]**.

❑ /^transp

This merge mode is in a certain sense the opposite of *^isl* in that it requires the secondary category to be transparent in stead of closed. It specifies that the argument lists in the passive direction - leftward - both must be unaffected. Being lexically assigned is one of the options here. The left agendas of both categories are transferred to the resulting category. On the other hand, the active list of the secondary category is bound to be empty (emptied, for example). Since this is a rightward rule, this requirement provokes a rightward embedding structure $(\ldots X_1(\ldots X_i(\ldots X_n)..)..)$, where the right arguments of the secondary category must be met before the constituent itself is subject to merge. As a consequence, this merge mode induces a kind of head adjunction, in that the phrases introducing the primary and the secondary head types end up as neighbors in the string.

(69) Prim \ u~LP / _~[Sec^*transp*|RP] $\otimes_{\text{transp}}$ Sec \ u~LS/ _~[ ] $\rightarrow$ Prim \ Fl~LS$\oplus$LP / a~RP

The secondary passive argument list ends up on top of the passive arguments brought in by the primary category. The secondary arguments will therefore be first to meet at the left hand side.
The typical case here is the cancellation of the infinitival verbal complement of a verb rasing (semi-)auxiliary. This was also the example given in (18). The particular appending of the passive argument lists under this merge accounts for the famous crossing dependencies in the Dutch verb cluster. In the following example, the np's are indexed to show this effect.

(70) *kan* 'can' *laten geven* 'let give'
s\u~[np_1^*isl*]/u~[vp^*transp*] $\otimes_{\text{transp}}$ vp\u~[np_2^*isl*, np_3^*isl*, np_4^*isl*]/a~[ ]
$\rightarrow$ s\ u~[np_2^*isl*, np_3^*isl*, np_4^*isl*, np_1^*isl*]/a~[ ]
kan laten geven 'can let give'.

This mode is a nice example of combining linearization and derivation. The emptiness condition on the secondary passive argument list forces this category to complete its right hand agenda before being involved in this merge. Unaffectedness marking on both left lists and append specification make the string instantiations of primary left arguments peripheral to the substring formed here and preceeding the string counterparts of the secondary arguments. Any string introduced to meet the needs of the primary category's left agenda is bound to occur to the left of the 'secondary' strings. Recall that no further operation can change that state-of-affairs. The only operations that the new argument list can be submitted to, are cancelling of its top element and/or appending. None of these operations changes the internal order established by */^transp*.

There is an alternative to (69) available. It allows for an alternative derivation in case the secondary category's left agenda is lexically empty, *i.e.* is empty and unaffected at merge. This would be its format:

(71) Prim \ _~LP / _~[Sec^*transpb*|RP] $\otimes_{\text{transpb}}$ Sec \ u~[ ]/ _~RS $\rightarrow$ Prim \ Fl~LP / a~RS$\otimes$RP

The primary left agenda is no longer required to be unaffected. The primary category may have consumed one or more of its left arguments before entering into this merge. This would not harm the 'crossing dependency' effects here, since the merge presupposes that there are no left arguments to the secondary category. Moreover, it defines an append at the right hand side that guarantees that the secondary categoty's arguments are first met. At this side this does not amount to crossing dependencies, but to completion of the secondary agenda before that of the primary right agenda. That is, in effect, the same result as was reached more rigidly by the emptiness requirement on RS in (69). Thus, (71) defines a derivationally liberalized subcase of, for example, verb clustering. It is not needed, however, to compute the verbal complex of Dutch in a syntactic and semantic adequate way.

❑ /^open

Another, structurally more intriguing, variety of /^transp drops the unaffectedness constraint on the secondary left agenda. Consequently, it allows the secondary category to consume part or all of its left agenda without destroying its fitness for this merge. A string associated with the primary category can therefore connect to a string , the left edge of which was not lexically associated with the secondary category.

(72) Prim \ u~LP / _~[Sec^*isl*|RP] $\otimes_{\text{isl}}$ Sec \ _~LS/ _~[ ] → Prim \ Fl~LS⊕LP / a~RP

Note that the (remaining) left arguments of the secondary at merge have to be put on top of the primary ones, which are aupposed to be unaffected. This merge does not mix up the left arguments differently, but puts less restrictions on the pre-merge behavior of the secondary category.
The canonical example in Dutch for this merge is the complementation of verbs which go into the such called 'third construction' (Den Besten XXX, among others). Prominent among them is *proberen* 'try', which is found to occur in each of the following, semantically equivalent, patterns. All other scramblings are out.

(73) …. dat jij probeerde Ramses Egyptian Nights te laten lezen
…. *'that you tried Jan Egyptian Nights to make read'*
(74) …. dat jij Ramses probeerde Egyptian Nights te laten lezen
(75) …. dat jij Ramses Egyptian Nights probeerde te laten lezen

Here is the instance of */^open* that would give rise to (74):

(76) probeerde Egyptian Nights te laten lezen
s\ u~[np$_1$^isl] / u~[vp^open] $\otimes_{\text{open}}$ vp\ a~[np$_2$^isl]/u~[ ] → s\a~[np$_2$ ^isl, np$_1$^isl]/a~[ ]
probeerde Egyptian Nights te laten lezen

The value for affectedness of the secondary right argument list results from two empty lists being appended at the previous merge, according to computation (66.4).
This merge mode is also basic to the arguments of adjunctive automorphisms. Verbal and sentential qualifiers may occur in a lot of positions inside verbal complexes. Consider the variety of positions for the instrumental adjunct *met een hamer* in the follwing sentences.

(77) … dat Henk met een hamer Jan de kast wilde laten openen
that Henk with a hammer Jan the cupboard wanted make open
(78) … dat Henk Jan met een hamer de kast wilde laten openen
(79) … dat Henk Jan de kast met een hamer wilde laten openen

In section XXX it will be argued that this flexibility reflects the combinatorial potency of *proberen*, making this merge mode to an essential ingredient of a categorial list grammar of Dutch.

❑ /^penins

The next mode stands accounts for the merge with near islands, *i.e.* constituents that have on their passive (leftward) agenda at most one, specified argument. In practice, this merge is used to account for combinations with constituents that lack fronted or leftward dislocated elements. A constituent licenses a dislocated element *iff* its left argument list contains a *type^mode* pair *xp^w*. The rule comes in two mutually exclusive formats, differing only in the specification of secondary passive (leftward) list. This double format amounts to a disjunction of specifications.

(80) Prim \ _~LP / _~[Sec^*penins*|RP] $\otimes_{\text{penins}}$ Sec\ _~[]/_~[] → Prim \ Fl~LP / a~RP
(81) Prim \ _~LP / _~[Sec^*penins*|RP] $\otimes_{\text{penins}}$ Sec\ _~[_^*w*]/_~[] → Prim \ Fl~LP⊗[_^*w*]/ a~RP

It is grammatically very important that in (81) the append in the passive direction suppresses the cancellation of the ^*w* argument in favor of cancellation of the primary category's left arguments. In this way, the intended dislocated element can only absorbed as the final act of completing the left agenda. That is also what the merge modality \^*w* will specify as a requirement. The example below illustrates a case of longe distance dislocation, where an argument of a deeply embedded infinitival complement is 'wh-ed', as in (83).

(82) *dwong* 'forced' *een boek te geven* 'a book to give'
s\u~[np^*isl*]/u~[vp^*penins*] $\otimes_{\text{penins}}$ vp\a~[np^*w*]/u~[] →
s\a~[np^*isl*, np^*w*]/a~[]
dwong een boek te geven
(83) wie zei henk dat agnes hem dwong een boek te geven
who said henk that agnes him forced a book to give
'who did henk say that agnes forced him to give a book to'

In order to assure that in this cases the ^*w* argument is properly transferred under composition, all appendings of left argument list - independently of the particular merge mode they figure in - should be defined such that ^*w* arguments are stacked down. To see why, consider the case of /^*transp* (69). If one of the complement's arguments is of the dislocated breed, this argument has to be surpressed in the output with respect to the arguments of the primary category. Otherwise, it would be at the top of the stack for at the wrong moment in the derivation. That is, we would like to be the output of the crucial ^*transp* merge in (84) to be as in (87), rather than as in (86) following (69).

(84) wie denk jij dat ik agnes een boek kan laten geven?
who think you that i can let play
'who do you think that I can let play'
(85) *kan* *laten geven*
s\u~[np^*isl*]/u~[vp^*transp*] $\otimes_{\text{transp}}$ vp\u~[np^*isl*, np^*isl*, np^*w*]/a~[] →
(86) → s\u~[np^*isl*, np^*w*]/u~[ ]
(87) → s\u~[np^*w*, np^*isl*]/u~[ ]

We can make sure that every ^*w* starts up at the bottom of its stack in the lexicon. Therefore, we only have to change the standard append, applied to argument lists, in such a way that ^*w* arguments at the bottom of either stack remain there. This is relevant for the tail of the list that is bound to become the upper part of the stack. An adequate reformulation of append in a Prolog like fashion may be this:

(88) clg_append(L, R, LR') if append(Lmin, [X^*w*], L) ,
append(R, [X^*w*], R') and append(Lmin, R', LR');
otherwise clg_append(L, R, LR) if append(L,R.LR).

From now on, append on argument lists is considered to be thus reformulated as clg_append. Note that (88) does not interfer with the alleged context freeness of this operation.

❑ /^transpnl

This mode is of limited use in the Delilah grammar. It is a variant of /^*transp*, differing only in the additional specification that the secondary active list must be affected. Consequently, no lexical category can comply with the secondary requirements of this mode. Its output conditions are the same as (69).

(89) Prim \ u~LP / _~[Sec^*transp*|RP] $\otimes_{\text{transp}}$ Sec\u~LS/a~[ ] $\rightarrow$ Prim \ Fl~LS$\oplus$LP / a~RP

Just like */^transp* this merge mode is mainly applied to deal with the verb cluster. This particular variation is partly accounting for the such-called *infinitivus-pro-participio* effects. An auxiliary verb may select a participle as the head of its verbal complement, but may instead require an inifinitive when the complement is verbally complex. Here are some relevant data.

(90) Hij had die baan wel gewild
he had that job wanted 'he would have wanted that job'
(91) Hij had die baan wel willen hebben
he had that job want to have 'he would have wanted to have that job'
(92) * Hij had die baan wel willen
he had that job want
(93) * Hij had die baan wel gewild hebben
he had that job wanted have

The mode */^transpnl* is assigned to the arguments of verbs that are sensitive to the *ipp*-effect. Unfortunately, the secondary active argument list being affected is not the whole story here. It appears that the argument cancelled in the construction of the secondary category must be vp, rather than anything else:

(94) Hij had gewild dat ik er bij was
he had wanted that I was present
(95) * Hij had willen dat ik er bij was
he had want that I was present

In section XXX the details of the grammarization of *ipp* will be reconsidered.

❑ /^qual

This mode requires the secondary passive list to be flagged *w*. This flag results from application of merge mode \^*wh*, to be discussed below. It thereby indicates that the last left mode dealt with a cancellation under this merge. The present mode consumes that flag.

(96) Prim \ u~LP / _~[Sec^*qual*|RP] $\otimes_{\text{qual}}$ Sec \ w~[ ]/ _~[ ] $\rightarrow$ Prim \ u~LP / a~RP

A canonical application is in the first category of the pair of categories to which an argument *wh*-phrase is assigned. A word like *wie* 'who' is assignes to two co-occurring categories. The rightmost of these is simply *np* or *np\u~[ ]\u~[ ]*, dealing with the selectional properties of the word as an argument to some other phrase in the sentence. The leftmost category that comes with *wie* selects the sentence that has been built by the merge that evaporated the rightmost category, and qualifies it. It may be qualified as a question, a postnominal modifier, and so on. Here is the example of *wie* introducing a question.

(97) q\u~[ ]/[s^*qual*]
np\u~[ ]/u~[ ] $\otimes_{\text{wh}}$ s\u~[np^*wh*]/u~[ ] $\rightarrow$ s\w~[ ]/u~[ ]

wie	slaapt
who	*sleeps*
q\u~[ ]/u~[ ]	

When the *wh*-term is not an argument, it comes with a single category, and this category cannot expect there to be a particular s. For example, *hoe* 'how' just qualifies its sentential argument. It requires that argument to have an unaffected left argument list, thus enforcing that the last and decisive merge of this argument was not leftward - compare the flag computation (66). The relevant mode is given below; it is as specific as (96) and therefore an exclusive alternative to it.

(98) Prim \u~LP /_~[Sec^*qual*|RP] $\otimes_{\text{qual}}$ Sec \u~[ ]/ _~[ ] → Prim \u~LP /a~RP

❑ /^adj

The last right merge is a very tolerant one: the primary category does not affect the status of the secondary category. The latter imposes all its specifications on the output. Even the output flag in the active direction is dictated by the secondary category, instead of being default *a* or *w.* The combinatorial and derivational effect of this merge is almost zero. The mode is particular suited for those merges which do not rise from selection or subcategorization, but rather from adjunction and modification. In these circumstances, the head types of the two categories wiil be equal, *Prim* = *Sec*.

(99) Prim_~LP/_~[Sec^*adj*|RP] $\otimes_{\text{adj}}$ Sec_~LS/Fr~RS → Prim\
Fl~LP⊕LS/Fr~RS⊕RP

A standard case would be the merge between a preposition introducing a verbal adjunct and the verb:

(100) …. vp\u~[r^*wh*]/u~[vp^*adj*] $\otimes_{\text{adj}}$ vp\u~[np^*isl*]/u~[ ]
mee *geschreven*
→ vp\u~[np^*isl*, r^*wh*]/u~[ ]
(waar heeft hij het boek) mee geschreven?
'(what has he the book) with written'
what has he written the book with? With what has he written the book?

The specific order in the left output list follows from the redefinition of append(88). The general picture here is that we need some 'invisible' merges, since not all combinatorics is by licensing.

This completes the set of rightward merge modalities, activated in the grammar of Delilah Dutch. There are seven of them, one of which, to wit */^transpnl,* is merely added to handle a particularity of Dutch and West-Germanic syntax. The other six represent core modalities, in that they represent essential combinatorial processes in Dutch sentence formation. Some of the rightward merges that are discussed below, are labeled like left directional merges. Except for the island-modalities, which represent rigid cancellation without any transfer of combinatorial agendas, it is only apparently the case that they are directional images. In fact, there is little symmetry in the selection of merges to deal with Dutch syntax. Linearization itself is the source of asymmetry.

1.1.2.2.2.3. Leftward Rules

It is seductive to call leftward cancellations backward, and rightward merges forward, following the terminology first used by Ades & Steedman (1980). It is also misleading , however, as it seems to refer to the parsing process in stead of to the linear restrictions on sequence formation. No choice shave been made explicit as to the way the grammar must or can be parsed. The point at stake here is incrementality in the linguistic interpretation of a sentence. In a non-flexible framework like the one adopted here, that is a point quite different from the question whether a certain configuration is left or right-headed.

❑ \^isl

The most rigid form of leftward cancellation indeed has its rightward image. The arguments cancelled under this mode, are basically of the same types as the ones which are submitted to /^*isl* : nominal phrases, determiner phrases, quantifiers. The standard case is the object arguments of verbs. Their canonical position of these objects is to the left of the verb, if Dutch is to be seen as an SOV-lookalike.

(101) Sec_~[ ]/_~[ ] $\otimes_{isl}$ Prim_~[Sec^*isl*|LP]/_~RP →
Prim\a~LP/Fr~RP

❑ \^transp

This mode is labelled like /^*transp* since it applies in order variants of configurations in which /^*transp* is invoked. Nevertheless, it imposes different types of input conditions. The primary category is forced to cancel its left argument first, whereas the secondary category is bound to be unaffected in both directions, *i.e.* to be lexical. The general formulation would be this:

(102) Sec\u~LS/u~RS $\otimes_{transp}$ Prim_~[Sec^*transp*|LP]/u~RP → Prim\a~LS⊕LP/u~RP⊕RS

The canonical application for this merge is the case of such called modal inversion: certain (semi-)auxiliaries may take the head of their verbal complements to the left, leaving the rest of their - or that head's - complement, if any, to the right. Here are relevant examples with the modal auxiliary *kunnen*.

(103) … dwingen kan te laten werken
… *force can to let work* 'can force (…) to make (…) work'
(104) … kan dwingen te laten werken
(105) … * dwingen te laten kan werken
(106) … * dwingen te laten werken kan

The relevant merge in constructing (103), then, is:

(107) *dwingen* *kan*
vp\u~[np$_1$^*isl*]/u~[vp^*open*] $\otimes_{transp}$ s\u~[vp^*transp*, np$_2$^*isl*]/u~[ ] → s\a~[np$_1$^*isl*, np$_2$^*isl*]/ u~[vp^*open*]
dwingen kan

Since primary categories introducing this mode will generally be (semi-)auxiliaries with just one complement, we may restrict the mode to situations in which the primary right argument list is empty, except for the verbal complement. Format (102) boils down, under these assumptions, to:

(108) Sec\u~LS/u~RS $\otimes_{transp}$ Prim_~[Sec^*transp*|LP]/u~[ ] → Prim\a~LS⊕LP/u~RS

❑ \^open

The most liberal of all merges imposes no input restrictions whatsoever. As such it belongs to the small class of modes with these characteristics, differing from each other only in the output specification.

(109) Sec\_~LS/_~RS $\otimes_{open}$ Prim\_~[Sec^*open*|LP]/_~RP → Prim\\1~LS⊕LP/Fr~RS⊕RP

The standard application here is to the automorphistic argument of an adjunct. In that case, however, we might as well choose for some more restrictive version. For example, we could consider the primary category to be 'closed' itself, in having no arguments unsaturated except the one to be cancelled. Such a restriction would imply that the primary category itself represents an island; this is, indeed, generally considered to be an identifying feature of adjuncts. Here is this more restricted version.

(110) Sec\_~LS/_~RS $\otimes_{open}$ Prim\_~[Sec^*open*]/_~[ ] → Prim\\1~LS/Fr~RS

Since the flag of an empty list hardly charges the relevant flag in the output, this mode lays the whole burden of output on the secondary category. All output lists stem from the secondary category and the passive flag may also be equal to the passive input in the secondary category. The dominance of the secondary category, again, seems to be in accordance with the nature of adjunctival modification. The primary category is a mere intervener or intruder into the secondary structure. In the following example, *heeft* 'has' would introduce the secondary, and *waarschijnlijk* 'probably' the primary category to an \\^*open* merge.

(111) Elke student heeft waarschijnlijk de boeken willen verkopen
every student has probably the books want sell
'every student probably wanted to sell the books'

For the sake of generality, however, we will stick with the liberal (109). It is subsumed by the more restrictive (110).

❑ \\^wh

This is the rather basic merge mode that cancels a leftward-dislocated complex against its licensing argument. It is supposed to be the final move in the completion of an argument structure. At every previous merge, the argument with this mode is oppressed, in the sense of kept at the bottom of the output stack, according to the revised definition of list append as clg_append in (88). Thus, there is no remaining active list in the primary category. The merge is designed to consume elements that in other frameworks may be considered to live in the specifier of a complementizer phrase. At output, the left argument list is flagged *w* for being affected by this particular cancellation.

(112) Sec\_~[ ]/_~[ ] $\otimes_{wh}$ Prim\_~[Sec^*wh*]/_~RP →
Prim\\w~[ ]/Fr~RP

The emptiness of the secondary active (leftward) list seems mandatory. Since this category is assumed to represent a left edge element, it must be completed at that side. The condition that the secondary passive list be also empty, amounts to the claim that only arguments can be cancelled this way, not heads. In this sense, the condition reflects the percolation of a *wh*-marking to some maximal projection, inducing pied piping.
Applications of this merge mode comprise standard wh-movement phenomena, topicalization and subject lefting. These structures are exemplified in the following set.

(113) (q\u~[ ]/u~[s^*qual*])
np\u~[ ]/u~[ ] $\otimes_{wh}$ s\u~[np^*wh*]/a~[ ] → s\w~[ ]/u~[ ]
wie *denk jij dat werkt* *wie denk jij dat werkt*
'who think you that works'
who do you think works?

(114) np\u~[ ]/u~[ ] $\otimes_{wh}$ s\u~[np^*wh*]/a~[ ] → s\w~[ ]/u~[ ]
die man *heb ik zien slapen* *die man heb ik zien slapen*
'that man have I see sleep'
that man I have seen sleeping

(115) np\u~[ ]/u~[ ] $\otimes_{wh}$ s\a~[np^*wh*]/a~[ ] → s\w~[ ]/u~[ ]
ik *slaap* *ik slaap*
'I sleep'
I am sleeping

In (113) it is assumed that lexical *wh*-elements come with two categories, rather than one (see also at /^*qual* above). That is the main difference between these elements and the dislocated entities in (114) and (115). The relevant merges as such are equal.

- \^adj

This is the leftward counterpart to /^*adj*. It has the same motivation and application.

(116) Sec\F~LS/_~RS $\otimes_{adj}$ Prim_~[Sec^*adj*|LP]/_~RP →
Prim\F~LS⊕LP/Fr~RP⊕RS

Here is a characteristic example:

(117) … s\u~[ ]/a~[ ] $\otimes_{adj}$ s\u~[s^*adj*, r^*wh*]/u~[ ]
zaten zij *op*
→ s\u~[r^*wh*]/u~[ ]
(daar) zaten zij op
' (that) sat they on' on that they sat

- \^part

Finally, in the rightward division, we need the inverse of an adjunct merge: a selected item the cancellation of which as a secondary category does not provoke affectedness.

(118) Sec\u~[ ]/u~[ ] $\otimes_{part}$ Prim\u~[Sec^*part* |LP]/_~RP →
Prim\u~LP/Fr~RP

There are all kinds of restrictions on the secondary input, as its main application is to lexical particles that may occur separated from their licenser. Those particles behave as if they are part of that licensing phrase, in that they may occur in positions in which other arguments to that licenser can not. The example below is with a resultative small clause predicate - such a predicate may occur with almost any transitive verb. It can take any position in cluster where it is accesible as the first-to-meet left argument to the verb, independently of verb-raising configurations.

(119) … dat Agnes Henk de auto groen wil laten verven
… *that Agnes Henk the car green wants make paint*
… 'that Agnes wants to make Henk paint the car green
(120) … dat Agnes Henk de auto wil groen laten verven
(121) … dat Agnes Henk de auto wil laten groen verven

The relevant merge for (121) is

(122) ap\u~[ ]/u~[ ] $\otimes_{\text{part}}$ vp\u~[ap^*part*, np^*isl*]/u~[ ]
groen *verven*
→ vp\u~[np^*isl*]/u~[ ]

Because the *u*-flag at the left argument list remains in the output, the resulting complex can go into the verb clustering required by the */^transp* mode on the complement of *laten*.

This completes the set of leftward merges. Again, the number is very restricted. One of the six deals with a local inversion, the subtle auxiliary switch in the Dutch verb cluster. The other five represent more or less core merges in the syntax of Dutch.

1.1.2.2.2.4. Dealing with adjuncts

In the exposition above, several modes were introduced that aim to deal with adjunctive modification. The nature of this modification is automorphistic. An adjunct sends its argument to the same category that the argument comes with. Consequently, adjunctive modification does not contribute essential ingredients to wellformedness or saturation. Moreover, adjunctive phrases are not very picky. They are usually capable of modifying diffirent sorts of phrases, thus taking different sort of categories as secondary. In doing so, they hardly impose conditions on the derivational status of their arguments. Finally, adjuncts are not subject themselves to discontinuity, peripheral to the sentential backbone as they are. They are rigid islands.
In CLG, then, it is not very difficult to think of a category for an adjunct. For example, a lexical adjunct that can occur as the left modifier to a verb phrase - say: *snel* 'quick(ly)'- is categorizable as follows:

(123) vp\u~[ ]/ [vp^*adj*]

The */^adj* mode is defined liberal enough in (99) to allow for all kinds of interference between this category and a constituent headed *vp*. Along (123), the word *snel* would also have to be assigned to categories taking sentences into sentences, noun into nouns, and so on. This would yield a restricted class of assignments - recall that because of merge modes there is no need to come up with different categories for different arities of verbs, not even when immediate (left) adjunction to these verbs as heads of vp has to be enabled. The class would be unsatisfactory, nevertheless, since it is not very general and would have to be repeated all through the lexicon. As an alternative, adjunctive phrases may be assigned to a categorial classifier, to be instantiated at need in the combinatorial process. Such a classifier would look like (123) but with a variable typing. The variable is supposed to be valuated in a closed finite class of types, *e.g.* {vp, s, n}.

(124) X\u~[ ] /u~[X^*adj*]

Subsequently, a transfer rule is needed to give, in a certain environment, the relevant instantiation. As a matter of fact, since not all automorphic merges behave exactly alike, we even could parameterize the merge mode in assignment (124), having target type and merge mode specified in one rule, yielding the category

(125) X\u~[ ] /u~[X^*M*]

Derived merge rules will have this format:

(126) X\u~[ ] /u~[X^*M*] $\otimes_M$ vp\Fl~LS/Fr~RS $\rightarrow$ vp\Fl~LS/Fr~RS only if
vp\u~[ ]/ u~[vp^*adj*] $\otimes_{adj}$ vp\Fl~LS/Fr~RS $\rightarrow$ vp\Fl~LS/Fr~RS
(127) X\u~[ ] /u~[X^*M*] $\otimes_M$ s\Fl~LS/Fr~RS $\rightarrow$ vp\Fl~LS/Fr~RS only if
s\u~[ ]/ u~[s^*isl*] $\otimes_{isl}$ s\Fl~LS/Fr~RS $\rightarrow$ s\Fl~LS/Fr~RS

It must be stressed that following this track does not affect the outline of the grammar as given in 1.1.2.1 and 1.1.2.2.1, but merely abbriviates lexical specifications.

1.1.2.3. The grammar of discontinuity

1.1.2.3.1. THE PARADOX: (DISCONTINUITY AS THE ENGINE OF LANGUAGE /) DISCONTINUITY AS THE CONSEQUENCE OF COHERENCE

Language chains words in meaningful manners. One word invokes another, some words justify themselves. The most intriguing fact about these chains is that they are multidimensional: phrases connect behind and over other phrases. Natural language hardly complies with the Polish notation. Phrases that determine each other's appearances and interpretations may occur side by side, or be separated by other elements which do not or little or less contribute to the connection. But then, a sentence is not just a chain but a bundle of ties, and connections may be indirect.
In order to get at a more explicit formulation of discontinuity: suppose that, in a given sentence S, we are able to determine for each word on which other word or phrase it depends. Dependency may develop along its morphological (as to its particular form), syntactic (as to its being licensed to occur) and semantic (as to its interpretations) dimension. We therefore must presuppose linguistic analysis. Let us moreover presuppose that every word has exactly one licenser in each dimension. This is not evident, but the point will return later on. The assumption allows for several licensers, although in different dimensions of dependency. Multi-dependency may be the normal case. It is widely assumed, for example, that the subject of a sentence in which tense is absorbed by an auxiliary, semantically relates to the main verb but morphologically and syntactically is determined by that tense bearing auxiliary:

(128) Ik heb enkele fouten begaan
I have some mistakes made 'I made some mistakes'

From a semantic point of view, then, (128) is discontinuous. The semantic or thematic content of the subject and therefore of the core predication has to be established over at least one phrase (*heb*) that does interfere with that content. It is for this reason that in most generative analyses of Dutch the subject is assumed to have moved from a position close to the main verb, to the left periphery. In these generative analyses movement expresses multiple loyalities, up to the necessitation of movement for reasons of interpretation - see *e.g.* Barbiers 1995.
Concentrating on this semantic dependency -which is after all the ultimate yield of sentence processing- let us name the semantic licenser of a word W *semlic(W)*. A first approximation of a semantically discontinuous string would then be:

(129) a sentence S is semantically discontinuous iff it contains a substring of non-empty phrases *X Y semlic(X)* or *semlic(X) Y X* such that neither *X* nor *semlic(X)* is *semlic(Y)*

This formulation entails that the element Y really is an intruder in the sense that it does not belong to the semantic domain of one of its neighbors. The typical case here would be any adjunct to X or semlic(X). The normal pattern would be, like in (128), that the intervener is itself a licenser of any of its neighbors. Suppose *heb* can be analyzed as the semantic licenser of *enkele fouten begaan*. The question, then, would be if semantic licensing is transitive, or if the property of being a semantic licenser percolates upwards under wrapping. This is a possible construal, but not a very enlightening one. It would amount to every phrase being an adjacent licenser. All discontinuity would be wrapped away. We may, however, gain structural insight by considering licensing to be intransitive and asymmetric.
In this vain, the approach to discontinuity of (129) suggests that interveners which are not selected by any neighbour, come with a licensing licensing mission themselves. What was said to be the normal case above - an intervener being linked to one of its neighbors - is also the 'cheapest' form of discontinuity. It is very unlikely that language has licensing nets of type *a b semlic(a) semlic(b)* without any pair of neighbors being related. Such a structure would be *disconnected*, and that is exactly what appears to be beyond sentence coherence.
To the extent that categorial grammars are able to express language coherence, the typological counterpart to *a b semlic(a) semlic(b)* would also be of a weird nature:

(130) a b c\a d\b

Here it is assumed that a negative occurrence of a type expresses being selected. In the realm of Lambek calculus, even strong hypothetical reasoning would not cope with sequents like these, since no proposition $a\ b\ c\backslash a\ d\backslash b \rightarrow x$ can be derived for any product-free type x . When no hypothetical reasoning is available as an instrument of grammatical analysis, sequents of this sort can, of course, not be derived. So it seems that the very mechanism of categorial combinatorics resists dependency networks that are merely crossing without being connected. Connectedness is the main engine of categorial combinatorics. This easily complies with free living specimen of crossing dependencies, like the interweaving of verbs and their selected objects in the Dutch verb cluster. Some examples were discussed above, and here is another one.

(131) … dat beren geen boeken willen lezen
… *that bears no books want read* 'that bears don't want to read books'
… … *beren geen boeken* semlic*(beren)* semlic*(geen boeken)*

The last line gives the primary selectional and semantic dependencies. But *willen* is not only the selectioner of the subject *beren*, but also of its verbal complement *lezen*. The last line of (131), thus, rather be like this:

(132) … beren geen boeken *semlic*(beren + *semlic*(geen boeken)) *semlic*(geen boeken)

Though *willen* really intervenes between *geen boeken* and *lezen*, it does so by getting connected. In fact, getting connected is such an evident feature of string formation in natural language, that one could easily overlook its profoundness. Crossing dependencies in Dutch appear to exist only under conditions of connectedness. That is not a trivial fact. The pattern established below could not occur, under these assumptions.

(133) * … dat boeken hem ik lezen (te) laten proberen wil
… *that books him I read make try want*

This configuration has both the order of the verbs and the order of the objects inverted with respect to the normal

(134) … dat ik hem boeken wil proberen te laten lezen
… *that I hem books want try to make read*

The order of the verbs in (133) reflects the standard order in the German verb cluster, the such called 'green' order. There are no languages, however, that mix up green verb order and crossing dependencies. (133) represents the impossible, at least the unobserved. A good reason for its rareness is the absence of connectedness between an intruding element, say the verb *lezen*, and its neighbors *ik* and *laten*, while that element itself is separated from its argument. The same would be true of the even more weird

(135) *… dat hem boeken ik laten proberen lezen wil
… that him books I make try read want

Though crossing here is painstakingly perfect, connectedness is gone. And one would hardly expect to find some Dutch-German interbreed establishing this pattern. It is hardly accidental that the only verbs that may select verbal complements to the left maintaining crossing dependencies, are some of those that do not introduce semantically dependent phrases in their own right, *i.e.* auxiliaries.

Crossing dependencies also represent discontinuity, in one of its crudest forms. It makes sense, then, to say that connectedness outbalances discontinuity. One may even conjecture that it is connectedness that makes discontinuity viable under the conditions of sentential coherence. This discontinuity is just one of the merges that connectedness can handle, remaining within the limits to which connectedness can stretch.

1.1.2.3.2. DISCONTINUITY AND GRAMMAR

Categorial grammar is said (Jacobson 1992) to make explicit the lexical relationship between phrases, *i.e.* the way one phrase selects an other to form a new one. It is only natural to assume that selection or licensing will develop by adjacency: phrases that co-occur in space or time, one taking the other along. Categorial grammar in particular lives, combinatorially, on adjacency, as Steedman (1990) notes. In the substructural format categorial grammar has adopted, functional application and composition are defined on concatenative pairs. Nothing can come between. Nevertheless, natural language is felt to abound with disorders: elements appears in positions where they are not surrounded by the phrases that we think license their occurrence in that sentence and provide an interpretative frame for them. The generative enterprise deserves credit for making explicit that discontinuities format sentence structure. In the present minimalist emanation of the generative view, the distinction between *move* and *merge* stresses discontinuity as a language organ. Moreover, the formulation of *move* as a one-structure-operation, handling features in one local tree, makes clear that discontinuity neither is unbounded nor at random, but inherently determined by properties of the structure. The general idea is that *merge* builds structure, and *move* changes it, subsequently. This really is a restyling of the division of labor between context free phrase structure rules and transformations, as established in earlier stages of generativism.
It is noteworthy that subsuming discontinuity under *move* does not mean that discontinuity is defined independently. As a matter of fact, there is circular traffic. Discontinuity is marked by applying *move*, and if something moves, it is discontinuous.

Categorial grammar combinatorics does not favor a principal distinction between operations of move and operations of merge. In the first categorial approaches to wh-movement, for example, forms of type raising were proposed to handle the interpretation of leftward dislocated elements. Later (*e.g.*, Moortgat

1988) special binary extract and insert operators were introduced to cover these configurations. They were subject, partially, to the same logic that governed the operators taking care of continuous sequences. From the point of view of linearization, however, these operators were of the 'anywhere' kind: $A \uparrow B$ was interpreted as the set of those split strings *ww'* such that *wbw'* is in *A* if *b* is in *B*, but the category gave no information as to where the string was to be invaded. In recent extensions of multimodal categorial grammar, as outlined in Moortgat (1997), the toolkit is extended to such a degree that specialized indexed operators, exclusive to certain structural configurations, can handle discontinuous or permutative instances of linearization. Discontinuity is handled by specialized structural operations on marked structures. With these wide coverage formalisms, it is possible to translate the minimalist distinctions between *move* and *merge* into categorial terms, as was shown by Vermaat (1999).

Still, it is not decided that *merge* and *move*, *i.e.* continuous and discontinuous linearizations, really are different processes that need distinct treatments in a grammar. Scrutinizing the definitions of *merge* and *move* as presented in Stabler (1996) en putting technical details aside, we are led to the following main difference between the two operations. *Merge* deals with two trees, such that one occupies one position in the other tree. *Move* deals with two trees such that one is related to two positions in the other. *Move* projects one tree twice, *merge* projects two trees once. Distinguishing these two processes is only mandatory to the extent that not every subtree of a tree is doubly or multiply projected in the end. That is, if every tree that is 'placed' by *merge* would be doubly committed by *move*, one might as well come up with one single operation *mergemove* that performs the structure building and the position linking in one strike. In that case, discontinuous occurrence would be the fate of every phrase, in the exact sense that every phrase except the one associated to the top node, links two or more not necessarily adjacent positions in the structure. Moreover, as *move* would be bound to merge, all double linking would be specified locally. It would amount to the conception that the occurrence of every phrase in a sentence is backed by a chain: to be a constituent is to be the head of a chain. Of course this is not really far from having every configuration licensed by feature checking, *i.e.* by matching pairs of features which are introduced at different positions. Here is an attempt to formulate such a unified structure building operation *mergemove*.

Let every lexical element come with a three-leaf labeled stucture [$_X$ Spec [X Compl]] such that Spec and Comp have the same type. For example: an ordinary intransitive verb like *walk* would be assigned [$_{vp}$ NP [walk NP]]. Spec and Comp are supposed to be chained, bearing the same relevant features except for lexical content. At each lexical structure, the position where the argument is lexicalized - either Spec or Compl - is marked; we will use italics to do so. The head is lexical *per se*. So, the category of *walk* might be [$_{vp}$ *NP* [walk NP]]. The structure [$_X$ Y [H Y]], having no lexcicalizations at all, is supposed to be equivalent to just H.
Mergemove is defined on every pair of trees [$_X$ *Y* [.X .Y]] or [$_X$ Y [.X .*Y*]] and [$_Y$ Z1 [. Y. Z2]], where either Z1 or Z2 is the lexical position and the dots mark possibly intervening right branching structure. It is assumed, however, that the head of every structure is uniquely defined as the highest position equal to the top label. This has the same effect as the directional head marking in Stabler's (1996) formalization of minimalistic combinatorics. This yields the following definition of *mergemove*:

(136) Mergemove
[$_X$ *Y* [.X .Y]] + [$_Y$ Z1 [. Y. Z2]] $\Rightarrow$ [$_X$ Z1 [$_X$ *Y* [. X . [$_Y$.Y. Z2]]
[$_X$ Y [.X .*Y*]] + [$_Y$ Z1 [. Y. Z2]] $\Rightarrow$ [$_X$ Z1 [$_X$ Y [. X .[$_Y$.*Y*. Z2]]

At merge, the lexical content of the head of the selected structure is placed in the position marked in the selecting structure. To see how things would turn out, consider the following mini-grammar for a language over the alfabet {1,2,3,4}. Let the lexicon be the following set of structures:

(137) { [$_1$ *2* [1 2]], [$_1$ 2 [1 *2*]], [$_2$ *3* [2 3]], [$_2$ 3 [2 *3*]], [$_3$ *4* [3 4]], [$_3$ 4 [3 *4*]], [$_4$ e [4 e]] }

Clearly, 1 is supposed to select 2, 2 to select 3, and 3 to select 4. Next is the derivation of the number 4213.

(138) $[_1$ *2* [1 2]] + $[_2$ 3 [2 *3*]] ⇒ $[_1$ 3 $[_1$ *2* [1 $[_2$ 2 *3*]]]]
$[_1$ 3 $[_1$ *2* [1 $[_2$ 2 *3*]]] + $[_3$ *4* [3 4]] ⇒ $[_1$ *4* $[_1$ 3 $[_1$ *2* [1 $[_2$ 2 $[_3$ *3* 4]]]]]]
$[_1$ *4* $[_1$ 3 $[_1$ *2* [1 $[_2$ 2 $[_3$ *3* 4]]] + $[_4$ e [4 e]] ⇒ [e$[_1$ *4* $[_1$ 3 $[_1$ *2* [1 $[_2$ 2 $[_3$ *3* $[_4$ 4 e]]]]]]]]
(= $[_1$ *4* $[_1$ 3 $[_1$ *2* [1 2 *3* 4]]]])

The linearization of the italicized elements and the head position gives the desired result.
The choice of categories is far from trivial. Under the present lexicon, not every order of the digits is derivable, while all chains are nested, around 1. In particular, no digit word can be derived in which 1 and 2 are not adjacent. This is due to the selectional restrictions implemented in the categories, *i.c.* the choice to have 2 selected by the element that invariabtly projects the top node. To exclude some more orders, one can limit the lexicon. For example, by taking the category $[_1$ *2* [1 2]] out one excludes the derivation of any number representation in which 2 occurs to the left of 1, which is half of the original language. The extension and the nature of the language are thus completely localized in the lexical specification. The above implementation proves that one can consistently and non-trivially consider merge and move to be a single operation.

This notion of *movemerge*, each substructure being doubly declared in the three, complies with the basic cancellation configuration of categorial grammar. Van Benthem (1986) proves a simple but very meaningful lemma on Lambekian categorial grammar. It is called count-invariance, and it amounts to the observation that in deductive systems based upon Lambek's rules, sequents can only be derived if there is for every basic type a certain balance in occurrences of that type in the sequent can be established. It is induced by a particular way of marking type occurrences that has been generalized to the proof net construal of Roorda (1990), by new the main proof construction device in linear logic. Here are the original notions.

(139) Count Protocol
for each basic type x, $count_x(x) = 1$ and $count_x(y) = 0$ if y is a basic type and $y \neq x$,
$count_x(y/z) = count_x(y\backslash z) = count_x(y) - count_x(z)$
$count_x(y_1, ..., y_n) = count_x(y_1) + ... + count_x(y_n)$.

Because of this way of counting, a complex type x/y will be said to contain a positive occurrence x of type x and a negative occurrence $/y$ of type y. The main application of type count is in the next statement.

(140) Count Invariance for grammar G
if $Y \Rightarrow z$ is derivable in some Lambekian categorial grammar G, then for all basic types x, $count_x(Y) = count_x(z)$. By consequence, if z is a basic type, then for all basic types $x \neq z$, $count_x(Y) = 0$.

Count invariance as expressed in (140) is a property of some deductive systems but can also be seen as a defining characteristic of a family of such systems, since the count protocol is independent of particular deductive programs. Count invariance says that for a type to occur in a well formed sequent, it has to occur an even times and in balanced positions. It is not hard to see here a reflection of the bijection principle, which also requires a balance between bounders and boundees. Under this point of view, categorial dynamics incorporates *move* rather than *merge*. But then, categorial systems are apt to deal with discontinuities to the very extent that theu are expressible by *move*.

In section 1.1.2.1.3.4 it was already pointed out that categorial list grammar tends to make no operational distinctions between merging and moving, merging being the process that both provokes adjacency of related items and may bring about separation of inherently related phrases. Such a grammar must nevertheless account for the deterministic and reconstructive aspects of discontinuity. Deterministic here means that the position of somehow dislocated elements is generally determined. Reconstruction points at the fact that dislocated elements in well formed and interpretable configurations always can be linked to their licensers.

1.1.2.3.3. THE VARIETY OF DISCONTINUITY

1.1.2.3.3.1. Sources of discontinuity in CLG

Above, in (129), discontinuity was introduced as a property of explicit relations between substrings, rather than as a structural possibility of the grammar describing those relations. The basic idea of that approach is that discontinuity arises by intervention of connected material, *i.e.* material that selects one of the separated phrases but is not selected by either phrase itself.
In CLG, phrases are related to categories that express their selectional condition. If a phrase occurs discontinuously, *i.e.* spacely separated from its selecting phrase, the nature of that discontinuity, as well as the nature of the interveners, must be expressed by the interaction of two other categories: that of the selecting and that of the intervening phrase. The nature of the interaction of the the two categories is given with the cancellation modality on the argument that complies with the head of the category of the selecting phrase. Here is an example with a right occurring selectioning phrase and a left adjuncting intervener. I is the intervening phrase, Sd the selected phrase, to be separated from Sg, the selectioning phrase. Cat_x stands for a category of phrase X. For the ease of exposure, let us assume that I is the only intervening phrase.

(141) Sd I Sg
Cat_{Sd} Cat_I Cat_{Sg}

$Cat_{Sd} = H_{Sd}\backslash Lf_{Sd}\sim L_{Sd}/Rf_{Sd}\sim R_{Sd}$
$Cat_I = H_I\backslash Lf_I\sim L_I/Rf_I\sim[H_{Sg}{}^\wedge Msg|R_I]$
$Cat_{Sg} = H_{Sg}\backslash Lf_{Sg}\sim[H_{Sd}{}^\wedge Msd|L_{Sg}]/Rf_{Sg}\sim R_{Sg}$

$Cat_I \quad \otimes_i \quad Cat_{Sg} \quad \rightarrow \quad Cat_{I+Sg}$
$H_I\backslash Lf_I\sim L_I/Rf_I\sim[H_{Sg}{}^\wedge Msg|R_I] \otimes_{Msg} H_{Sg}\backslash Lf_{Sg}\sim[H_{Sd}{}^\wedge Msd|L_{Sg}]/Rf_{Sg}\sim R_{Sg} \rightarrow$
$H_I\backslash Lf_{I+Sg}\sim([H_{Sd}{}^\wedge Msd|L_{Sg}]\oplus_{Msg}L_I)/Rf_{I+Sg}\sim(R_I\oplus R_{Sg})$

$Cat_{Sd} \quad \otimes_j \quad Cat_{I+Sg} \quad \rightarrow \quad Cat(_{I+Sg)+Sd}$
$H_{Sd}\backslash Lf_{Sd}\sim L_{Sd}/Rf_{Sd}\sim R_{Sd} \otimes_{Msd} H_I\backslash Lf_{I+Sg}\sim([H_{Sd}{}^\wedge Msd|L_{Sg}] \oplus_{Msg} L_I)/Rf_{I+Sg}\sim(R_I\oplus R_{Sg}) \rightarrow$
$H_I\backslash Lf_{I+Sg+Sd}\sim((L_{Sg} \oplus_{Msg} L_I)\oplus_{Msd} L_{Sd})/Rf_{I+Sg+Sd}\sim((R_I\oplus R_{Sg})\oplus_{Msd} R_{Sd})$

Note that the intervener determines the type of the merged phrase Sd+I+Sg. It is the primary category in the first merge, cancelling the head of the selectioning phrase, and it determines the primary category in the second merge, cancelling the head of the slectioned phrase. Abstracting from direction, this is the only relation between the phrases Sg, I and Sd that can be reconstructed by CLG and complies with the intuitive definition (129). Connectedness is realized in CLG as generalized composition, taking over

agendas under merge. Because connectedness is interpreted here as composition, the category Cat_I is not only intervening, but also bridging between Sd and Sg, *c.q.* their respective categories. Intervening, it brings the two occurrences of H_{Sd} - the negative and the positive one - together in adjacent positions, after all.
In CLG, then, discontinuity is accounted for to the extent that bridges exist or can be constructed. Connectedness has a clear cut categorial condition, in the form of a predictable construal, within the limits of finite combinatorics, of bridging interveners. The construal of discontinuity is dictated by the specification of the merge modes involved in the bridging, Msd and Msg in the example above. In particular, they may maximalize or minimalize the effects of discontinuity by the way they append argument list and by zero requirements on argument list of the categories involved. Clearly, the merge under mode Msg may introduce additional discontinuity if L_I, L_{Sg} or R_I is not empty: this merge will mark the phrase associated with the primary category, or the phrase associated with the secondary category or both as an intervener with respect to the arguments in these lists.
This construal of discontinuity in CLG is will be scrutinized in the next sections.

1.1.2.3.3.2. Toolkit of discontinuity

To get a clear idea of what discontinuity under CLG is up to, consider two phrases *P* and *S* - for primary and secondary string, respectively - and their respective categories P\ua~[LP^i]/ua~[S^j, RP_1 ^k] and S\ua~[LS^l]/ua~[RS^m]. Let *LP*, *RP*, *LS* and *RS* be phrases of categories that can be cancelled against the corresponding arguments in the categories of *P* and *S*. *P* and *S* stand for the smallest phrases with the corresponfing heda categories, *i.e.* for the phrases introducing the primary and the secondary head types, respectively. Depending on the specification of the modes *i, j, k, l, m*, any of the following strings may model this family of merges. P and phrases selected by it (*LP* and *RP*) are italicized for transparency.

(142) LS *LP* *P* S RS *RP*
(143) *LP* LS *P* S RS *RP*
(144) LS *LP* *P* S *RP* RS
(145) *LP* LS *P* S *RP* RS

Let us evaluate the nature and measure the extend of discontinuities arising in these strings and show that the strings are ordered by increasing discontinuity, the middle pair being equal in this respect. In (142) the strings LS and S are separated, whereas S is the selector for LS. The interveners are not themselves selected by these strings. This is real discontinuity, according to (129). LP and P are adjacent. So are S and RS. P and RP are separated again, but by a phrase and its selection that is selected itself by P; this does not count as discontinuity, since it simply reflects linear ordering of arguments. This string therefore shows one specimen of discontinuous concatenation.
In (143) both LP and LS are separated from their selecting phrase. None of the interveners is selected by the separated phrases; consequently, these are two independent discontinuities. At the right wing things are the same as in the first string. The total number of discontinuities therefore is two.
Compared to string (142), (144) has one additional discontinuity at the right. This brings the number of discontinuities here at two.
The last string shows discontinuities between every argument string and its selector. The number of discontinuities here must be four.
Several aspects of this exercise are noteworthy. First, none of the four strings is fully continuous. Consequently, every merge with four non-empty argument lists induces some form of discontinuity. Second, the number of discontinues pairs varies greatly, depending on the ways of appending (non-empty) argument lists.
Third, some discontinuities do not respect connectedness. In particular, the discontinuities between LP and P in (143) and (145) are marked in that the intervener, a phrase selected by *S*, is not related by selection to any of its neighbors. The same holds for the discontinuity of S and RS in (144) and (145).

Next is a graphical representation of the dependencies in the four strings. The arrow points at the selected phrase.

(146)
LS *LP* *P* S RS *RP*
(147)
LP LS *P* S RS *RP*
(148)
LS *LP* *P* S *RP* RS
(149)
LP LS *P* S *RP* RS

From this pictures one easily read that continuous linearizations come with outgoing arrows in the same direction. Connected discontinuities are marked by an incoming and an outgoing arrow in opposite directions. Unconnected discontinuity invoke crossing arrows in the same direction.

As was already conjectured in section 1.1.2.3.1, unconnected discontinuity may be avoided in natural language. Note that in the present casting this form of discontinuity is caused by intervention of 'dislocated' argument phrases. The grammar can exclude the occurrence of unconnected discontinuity, then, in either of two ways.
One way is requiring one or both of the relevant input argument lists to be empty. The other way is appending the lists in such a way that arguments cannot intervene between another argument and its selecting phrase. The latter way induces crossing dependencies. In that sense, crossing dependencies may result from a combinatoric strategy avoiding unconnected discontinuity. It also may be seen as resulting from a certain agenda management. Consider the left part of the sequence(143), now decorated with brackets indicating the intended order of cancellation.

(150) (*LP* (LS (*P* S)))….

If S introduces the first argument to be cancelled, its agenda must be on top of the merged category. It is a kind of democratic usurpation. Compare this configuration to the following bracketed version of (142).

(151) (LS (*LP* (*P* S)))

Although the cernel merge is the same as above, the agenda of the primary category must be on top to deal with the string after that merge. The category introduced by S is not only cancelled, but its agenda is also oppressed. If we consider these agendas as stacks, it means that at merge the secondary stack is to be put below the primary one. If we furthermore assume that the primary category is less affected by the cancellation than the secondary category is, this stack order does not comply with that intuition: the primary stack has, so to say, to be lifted in order to stay on top. Putting the secondary agenda on top, is less effort and more 'context free'.
A smoothier bracketing for (142) would be

(152) (LS (*LP* *P*) S))

Exactly this order of constituency can be reached by choosing a merge mode that imposes on the primary category the input condition that its passive argument list be empty. P and its left agenda are then one constituent at merge with the secondary category.

In this vain, crossing dependencies express the tight relation between the phrases P and S, instantiated by the early merge of their categories. Under this tight relation, appending such that connectivity is assured turns out to be the cheapest option.

Here are some instances of merge mode definitions to this effect, guided by the examples above. Only the elements relevant to the strategy are specified.

(153)	P_~[ ]/_~[S^j|_]	$\otimes_j$	S_~LS/_~_	→	P_~LS/_~_
(154)	P_~LP/_~[S^j|_]	$\otimes_j$	S_~[ ]/_~_	→	P_~LP/_~_
(155)	P_~LP/_~[S^j|_]	$\otimes_j$	S_~LS/_~_	→	P_~LS⊕LP/_~_

The two ways of avoiding disconnectedness are not independent of each other. Choosing the empty list strategy makes the append strategy obsolete. If the secondary argument list is deemed to be empty, as in (154), that category is sealed to cover an island. If the primary argument list is marked empty, this can only mean that the other arguments -which are required to be cancelled already- are more intrinsic to the head introducing phrase than the argument the cancellation of which is dealt with here.
Thus, once discontinuity is an option, crossing dependency is a normal configuration, resulting from a sound strategy of appending non-empty passive argument lists such that connectedness is assured.

Although unconnectedness can be avoided along two tracks, the ways are rather different. Imposing emptiness on one or more argument list at a merge, isolates a constituent in the relevant direction. It is partially 'islanded'. If a merge requires both argument lists of a category to be empty, the category represents a full island. It seems more natural to think about islands in terms of secondary categories than to think of islands represented by primary categories. A primary island is conceptually strange in that sense that it is hard to think of reasons why the cancelling of one particular argument would require all other arguments of that category to be cancelled already. It would impose an order of cancellation that at least within the argument list itself is already given with the order of the stack. The empty lists condition on the primary category can therefore only be relevant at the level of directional priority. So, such a full island restriction on the primary category amounts to an order of directional applications - all left arguments to be cancelled before the last right one, for example. This would be, however, a global restriction that has no direct effect on linear order. But linearization is the name of the game. Thus it is not surprising that none of the standard combinatory rules for Delilah's grammar of Dutch is of the following type:

(156)	P_~[ ]/_~[S^j]	$\otimes_j$	S_~LS/_~RS	→	P_~LS/_~RS

This type of restriction, however, is adequate for adjunctive primary categories, automorphisms of the type *x/x* or *x\x*. They are not very likely to get involved in complicated patterns of crossed or nested dependencies. They do not impose heavy conditions on their arguments. In fact, their categories only linearize the adjunctive phrases themselves. The argument the cancellation of which comes with this particular merge mode, is of a nature differing from the arguments which the merge modes requires to be cancelled in advance. An adjunctive preposition - in contrast to prepositions heading an argument to another phrase - takes a nominal element to become a modifier which may take all sorts of arguments automorphistically; for example:

(157)	*op, met, in, tussen*	→	vp\ua~[ ]/ua~[np^*isl*, vp^*prep*]

Although prepositions may be constructed as relations between the denotations of their two arguments, we will hardly consider them to modify their nominal complement. The automorphistic merge is special, as it seems. It requires the primary category to behave coherently and fully continuously. As will be explained below, it seems that at adjunction, the adjunct is not the usurping kind of primary category. Its input conditions seem to justify the conjecture made by Bouma and Van Noord (1994) that adjuncts may be selected themselves. In the present setting, this means that they would be selected and cancelled under the island mode.

Quite different is the situation with respect to the argument list of secondary categories in a merge mode. It seems reasonable to assume that every language has processes which are covered by the following merge mode, requiring the secondary category to be a full island; its an abstraction of */^isl* in 1.1.2.2.2.2.

(158) P_~LP/_~[S^i|RP] $\otimes_i$ S_~[ ]/_ ~[ ] $\rightarrow$ P_~LP/_~RP

In this - quite common - case the merge mode puts no restriction whatsoever on the primary category. The secondary category is usurped upon completely; its cancellation has no further influence on the remaining configuration. Of course, no discontinuity at all arises from this merge. Any empty argument lists of the secondary category will prevent discontinuity to arise in its direction.
As for the secondary passive list in rightward cancellation, we may observe a connection between derivation and discontinuity that resembles the reflections on the primary passive list following (153) - (155) above. Consider the following format for rightward cancellation under some mode *mod*.

(159) P\Plf~LP/Prf~[S^*mod*|RP] $\otimes_{mod}$ S\ Slf~LS/Srf~RS $\rightarrow$ P\Lf~(LP$\oplus_{mod}$LS)/Rf~(RP$\oplus_{mod}$RS)

At the right wing, requiring RS to be empty amounts to the same linearization as choosing an append such that RS is on top of RP. From a structural point of view, this append is therefore obsolete, since it would just influence the derivational order of cancelation but not the linearization. We might want, however, for some processing reason, to have the S argument cancelled before S cancels its own arguments. In that case, we must specify this particular append in the merge mode for S rather than requiring its passive list to be empty (and done). Recall that the merge modes are activated under directed application, and that any parsing strategy will implement some form of directionality too.

As for affectedness, it would be weird to have merge modes conditionalizing the active list of the primary category. Modes are activated define for the top argument in that list only, and requiring this list to have been or have been not operative before amounts to redefining the position of the top argument -the one to be cancelled- in that list. That makes no sense, as that order is defined by the lexicon and previous merges. Therefore, the flag on the active primary list will always be unspecified in the definition of a merge mode.

The following table contains an overview of the linearization effects of every possible specification. The condition in that table means that the specification has the effect mentioned only if that condition is fulfilled. Otherwise, it is without any effect on linearization.

(160) tabel of merge effects for tightward cancelling

specification	*in combination with*	*linearization/bracketing*	*linerization effects*
LP := [ ]		LS (P S .).	coherent discontinuity
LS := [ ]		LP (P S .).	continuity
RP := [ ]		.(.P S) LS	continuity
RS := [ ]		.(.P S) RP	continuity
left append := LS+LP	LP, LS ≠ [ ]	LP (LS (P S .).	discontinuity

left append := LP+LS	LP, LS ≠ [ ]	LS (LP (P S .).	discontinuity
right append := RP+RS	RP, RS ≠ [ ]	.(.P S) RP) RS	discontinuity
right append := RS+RP	RP, RS ≠ [ ]	.(.P S) RS) RP	continuity
Plf := ua	append = LS+LP	LP (LS (P S.).	coherent discontinuity
Plf := a	append = LS+LP	LP" (LS ((LP' P) S.).	incoherent discontinuity
Slf := ua	append = LS+LP	LP (LS (P S .).	coherent discontinuity
Slf := a	append = LP+LS	LS" LP (P (LS' S .).	random discontinuity
Srf := ua	append = RP+RS	.(. P S) RP) RS	incoherent discontinuity
Srf := a	append = RP+RS	.(.P (S RS')) RP RS"	random discontinuity

Discontinuity is dubbed 'incoherent' if intervening phrases are not selective with respect to the separated pair. This discontinuity arises in particular when a secondary selector is separated from its arguments by primary arguments, which are in no way related to the secondary selector or to its arguments.
Discontinuity is dubbed "random" if affectedness flags allow argument lists to be broken up, the parts being separated by unconnected phrases.
The above scheme suggests a hierarchy in specifications of merge modes affecting linearization. The strongest effect results from emptiness conditions on input argument lists. Any such specification guarantees continuity or coherent discontinuity at its side, without further specifications being necessary. Next are the ways of appending. Their effect is conditionalized by non-emptiness of any of the argument lists involved. Their effect, thus, is dependent. Finally, the effect of affectedness specification lives on the choice for particular appendings, which were already dependent themselves. Flags are thus the most subtle instruments to influence linearization. Appends bring in discontinuity, flags specify the nature of the discontinuity. Recall that flags are sensitive to derivational history, in ways specified in (66) for Dutch. A flag *ua* encodes that the list has not been activated till then, or that it was not involved in the last few merges.
The hierarchy makes some combinations obsolete or nonsensical, or just marked. For example, it is not easy to see what any appending or flagging can add to the empty input specification. In the same vain, flagging specifications only have effect when append is discontinuous. Moreover, it may seem that there may be more options to the same effect. Compare for example the following merge modes.

(161)
(161.1) P_~LP/_~[S^*i* |RP] $\otimes_i$ S_~LS/ua~RS → P_~LP $\oplus_i$ LS/_~RS+RP
(161.2) P_~LP/_~[S^*j* |RP] $\otimes_j$ S_~LS/a~[ ] → P_~LP $\oplus_j$ L/_~RP
(161.3) P_~LP/_~[S^*k* |RP] $\otimes_k$ S_~LS/_~[ ] → P_~LP $\oplus_k$ LS/_~RP

At first glance the linearization effects may appear the same: in each case the relevant order of phrases will be . . *P* S LS *LP*. The bracketings differ, however: .(.*P* S) LS) *LP* for (161.1), .(.*P* (S LS)) *LP* for the two other cases. Only the first specification assures the merge of the primary category with a lexical secondary category. (161.2) excludes a lexical status for the secondary category, whereas (161.3) is indifferent in this respect. The options thus vary in their derivational consequences.

1.1.2.3.4. PATTERNS OF DUTCH

The next table holds an overview of the specifications used in the almost-complete syntax of Dutch, presented in section 1.1.2.2.2. Instead of naming lists by direction, the table generalizes over directionality and names list (primary or secondary) passive or active, active being the direction of cancellation. Only if some explicit valued is required by the merge mode, this requirement is specified in the relevant column. To the extent that modes seem to have the same specifications, they may differ as to exceptional output flagging, which is not accounted for in this table. Recall, furthermore, that appending of left lists always implies down stacking of an wh-argument in either lists, according to provision (88).

(162) Table of merge specifications for Delilah's grammar of Dutch

P\PPf~PP/_~[S^*i* |PA] $\otimes_i$ S\SPf~SP/SAf~SA → P_~appendP/_~appendA

S\SAf~SA/SPf~SP $\otimes_j$ P_~[S^*j* |PA]/PPf~PP → P_~appendA/_~appendP

mode	*PP*	*SA*	*SP*	*PPf*	*SAf*	*SPf*	*appendP*	*appendA*
/^*isl*		[ ]	[ ]					
/^*transp*		[ ]		u		u	SP+PP	
/^*open*		[ ]		u		-/u	SP+PP	
/^*penins*		[ ]	[ ]/[x^*w*]				PP+SP	
/^*transpnl*		[ ]		u	a	u	SP+PP	
/^*qual*		[ ]	[ ]	u		w/u		
/^*adj*							PP+SP	SA+PA
\^*isl*		[ ]	[ ]					
\^*transp*	-/[ ]			u	u	u	PP+SP	SA+PA
\^*open*	-/[ ]						SP+PP	SA+PA
\^*wh*		[ ]	[ ]					
\^*adj*							PP+SP	SA+PA
\^*part*		[ ]	[ ]	u	u	u		

Inspecting this table, one can easily make some observations regarding the nature of the merges needed for this fragment of Dutch. Before going into details, however, it must be stressed that other grammars of the same fragment may be as efficient or more enlightening. One might even compare grammars with respect to the instruments they choose for determining the family of merge modes doing the job. Here is a report on the use of the discontinuity toolkit in Delilah's grammar.

What immediately comes into the eye is that almost all empty lists input specifications concern the secondary category. The only cases of such specification for a primary list are by way of an alternative formulation to a rule without sucg specification. These alternatives are driven by reflection, not by necessity, though. Thus, in this grammar the main load of discontinuity management is put upon the secondary category. This is the natural culprit, as it seems. At the same time, the primary category is often required to be unaffected in the passive direction and to allow the secondary passive agenda to be first to meet, thus reducing the chance of incoherent discontinuity. It is, furthermore, remarkable that in particular SA, which is that edge of the secondary category that is necessarily separated from the primary category, tends to be completed at merge.
None of the merges involves a requirement on non-emptiness of input argument lists. The possible specification that an input list be affected, is used only once, in the mode /^*transpnl*. This mode exclusively deals with *infinitivus-pro-participio* effects, which only occur in the presence of complex verb clusters. All other affectedness specifications require an input list to be unaffected.
As for appendings, it is noteworthy that the passive direction always complies with the side the primary category is on. Thus, from a directional point of view, the active appendings in the leftward cancellations all reflect the priority of the secondary left agenda, *i.e.* the imposition of continuity. Accordingly, the passive appendings in the rightward cancellations express the same priority, now invoking coherent discontinuity. The only exception here concerns a secondary left list that only contains a *wh*-element, this element being the only one to escape from the secondary domain.

As a whole, the grammar of Dutch according to this specification lives on only a few instruments of discontinuity management. The variety of means is rather limited.

1.1.2.3.5. COORDINATION AS DISCONTINUITY

Coordinated phrases are most certainly incoherently discontinuous. In the presence of coordinators, it is inevitable that at least one phrase will be separated from its selecetor or its selection by other phrases among which the non-connected coordinator. Coordinators have no special licensing relation with any phrase or category, and thus, they are incoherent interveners by definition. This is the case both for constituent and for non-constituent coordination.

(163) Ik heb haar echtgenoot en diens moeder *ontmoet*
I have her husband and his mother met
(164) Ik heb haar echtgenoot de fiets en diens moeder de auto *gegeven*
I have her husband the bike and his mother the car given
(165) … dat ik de jongens of jij de meisjes *moet begeleiden*
… that I the boys and you the girls should supervise

In the sentences above the selectors of the conjoined nominal phrases are italicized. One can easily see that they are necessarily separated from at least one of their selections by at least the conjunction itself. Nevertheless there are several good reasons to treat coordination as a structure of a nature different from other discontinuities. Delilah handles coordination by a distinct algorithm that is fed by the grammar and feeds it, but is steered by essentially different procedures and aims. The algorithm is introduced and accounted for in Cremers (1993), and will be discussed separately in section XXX. It sets coordination apart from the computation of other linguistic structure because the mechanics of these computations do not fit the nature and the scope of coordination. Here are the arguments for a dedicated treatment of coordinate structures.

1.1.2.3.5.1. Conjunctions don't select

There are only a few, mostly focus related restrictions on the occurrence of conjunctions and conjuncts in a sentence. The general rule is that if each conjunct can be properly contrasted with the other, the conjunction is ok. Wellformedness depends on the information contour rather than on the structural properties of the senetnce. A recipe to form coordinated structures could be this. Take any wellformed sentence without coordination. Choose any position to the right of some phrase. Take some string to the left of that position. Construct another string that is contrastive to the selected string, and insert the conjunction with the constructed string in the choosen position. The result will generally be a wellformed coordinated structure. The main exception to this recipe for coordination relates to the conjunction of - in quantifier terms -
singular filters generated by nonempty aroms; they may run into agreement violations under this procedure. But these are rather innocent, fully predictable and easily explained (see Cremers 1993: ch. 2). In any case it is difficult to maintain that conjunctions are biased towards certain types of phrases or categories. Equally difficult it may be to maintain that they select or license one or both of the conjuncts. The analysis in Johannesen (1993) to this effect, within standard generative categorial projection, fails on the non-constituent status of some conjuncts. On the other hand, conjunctions can be addressed as behaving similar to adjuncts, differing mainly in picking two instead of one arguments. The problem for this view is again that other adjuncts almost exclusively choose welldefined categories as automorphic targets whereas conjunctions handle almost any string of phrases independent of their categorial status. This unselective behaviour of conjunctions is reflected in the fact that from inspecting only one side of a conjunction, one cannot tell which is the conjoined subphrase. Being a conjunct is not an independent

configurational property of a phrase. I take this argument weighting heavily against any categorial characterization of conjunctions in languages like Dutch. Rather, they should be treated syncategorimatically, not subjected to the core categorial engines of grammar.

1.1.2.3.5.2. Conjunctions are not selected

There is a good argument that conjuncts play a role in the semantics of certain maximality expressions, made by Postma (1995): *met man en muis vergaan* ('with man and mouse sink'), *have en goed verliezen* (everything loose). One might even say that in these cases the conjunction is selected by the verb as the head of an expression of grade. It is also true, however, that all of these expressions are frozen. Notwithstanding Postma's observation that the conjunction introduces maximality here, it makes no sense to say that the conjunction is selected by the verb as a conjunction: except for one all conjunctions fail in this position. If so, there are no phrases at all that select conjunctions. Conjunctions don't occur at the demand side of the grammar. Note that conjunctions and adjuncts differ in this respect. It is transparent to reverse the normal automorphic analysis of adjuncts into an argument analysis in which adjuncts are selected, even with certain limitations, by other phrases. Analyses along these lines have forwarded by, *e.g.* Bouma en Van Noord (1994) and Janssen (1986??). No such move is conceivable for conjunctions. Inspecific as conjunctions are, it would simply double or more than double the set of categorial specifications, according to the formula: if category X is selected then also *X and X* and *X or X*. No analytical gain would result from doing so, however. This 'selection' of conjuncts is as inspecific and uninformative as conjunctions are general and uneclectic.

1.1.2.3.5.3. Coordinations do not obey count invariance

Count invariance (140) is a major distinctive feature of resource sensitive type logics and categorial grammars. It identifies the combinatoric engine of this systems as establishing a relation between two items fulfilling opposite roles in the structure. By definition, coordination entails the multiplication of certain roles in a configuration. In the presence of coordination the balance between selectors and selectees breaks down. Only by considering conjunctions as unselective automorphic selectors one can overcome this unbalance. This is effected by assigning them to type $(x\backslash x)/x$, where the two negative occurrences of x restore the balance against the two positively occurring conjuncts which they aim to cover. But then count invariance looses its edge: we cannot tell which is the balance to be restored before having parsed the sentence in order to identify the conjuncts that bring in the type doubling. And we must parse because the conjuncts themselves are not identified locally. This paradox of coordination is addressed in Cremers (1993) and Cremers and Hijzelendoorn (1997a). In the latter it is argued that the best counts we can get in the presence of coordination are (complex) inequalities, whereas count invariance is expressed in terms of equalities.

1.1.2.3.5.4. Coordinations are semantical non-constructs

Till now little reference has been made to what grammar is about: the conveyance of meaning. It is has been silently assumed that phrases have interpretations, and that whatever has a category, also has a meaning, in compliance with respected insights in natural language meaning construal, say compositionality (see for an excellent overview Janssen 1997). In chapter XXX categorial list grammar is projected in this field. In this paragraph I will only argue that conjunctions, *i.e.* the unit formed by the two coordinates and the coordinator, are not generally interpretable prior to the interpretation of the sentence or sentences which the coordination is part of.
First, it is wise to recall the statement from preceding paragraphs that the identification of the coordination is the result of parsing, not a subprocess of it, as the identification of any other constituent is. This alone already implies that the meaning of the conjunction cannot contribute independently to the meaning of the sentence that is construed from parsing it. But even if we abstract from this contradiction, and assume that coordinations contribute to the meaning construal by oracle, it is hard to determine what

that contribution could be. Suppose that we were, by oracle, capable of identifying any conjunction as some constituent of a certain category with structure *$[_{cat}$ cat conj cat]*. In order to establish a meaning for it, we must be able to identify an operation executed by the conjunct on the meanings of the conjuncts, yielding one 'merged' meaning. It is tempting to interpret *conj* as a certain operation defined on the algebra D_{cat} of denotations of category *cat.* It is well known, though, that conjunctions do not show boolean behaviour, in the sense that they denote a fixed algebraic operation on their arguments' algebra. The standard example is with quantifiers, *i.e.* denotations of type *np*. In sentence (166.1) the coordination is most likeley interpreted as the meet over the two quantifiers it conjoins: what is predicated of the coordination, is predicated over each of the coordinates. In (166.2), on the other hand, nothing is predicated of each of the coordinates. Many scolars have suggested that we need something like a group denotation to get the semantics in order here. In (167) it is rather the join over the two denotations that should be the coordination's contribution to the sentence meaning. The reason is quite clear. The coordination of (167) does not denote the set of spots on earth that are both Norwegian and Swedish, but rather the set of spaces that are (either) Norwegian or Swedish.

(166)
(166.1) De regering heeft [Smit en De Vries] bevorderd tot ridder in een of andere orde
'The government has Smit and De Vries promoted to knight in one or other order'
(166.2) De regering heeft [Smit en De Vries] belast met het onderzoek
'The government has Smit and De Vries charched with the investigation'
(167) In Noorwegen en Zweden rookt niemand meer
'In Norway and Sweden smokes nobody anymore'

What a coordinator should do semantically to its coordinates, may depend on functional properties of its environment, as was made abundantly clear by Zwarts (1986), or on denotational properties of the predication that the coordination is involved in (see Link ??? for the original argument). Even though this interaction is steered by laws of logic and algebra, this still implies that there is no independent contribution to the semantics of a sentence made by a coordination. Rather, the converse is the case: in order to define the semantic nature of the coordinative linking, we have to interpret the sentence which it is part of.
Janssen (1997) considers arguments against compositionality to this effect, as brought forward by Higginbotham (1986) with respect to the interpretation of *unless*. Here too one can observe that *unless* behaves different in different functional domains.

(168) Every person will eat steak unless he eats lobster
(169) No person will eat steak unless he eats lobster

In an upward entailing environmment as the nuclear domain of a universal quantifier, *unless* denotes disjunction. In a downward entailing environment it denotes conjunction. Janssen, who considers compositionality to be a requisite of grammatical architecture, rather than a decidable property of language, gives two ways out in this case: either *unless* is constructed as a function from contexts to values, or it is considered to be ambiguous. Suppose we would try to subsume the context dependency of the interpretation of constituent coordination under the same umbrellas. The first solution would not solve our problem, because there is no context available when we try to interpret coordination. *Unless* precedes or follows a proposition that may specify the relevant context. Constituent coordination is part of the context that should license its interpretation, since there is no proposition independent of the coordination. Moreover, coordinations may be part of each other's contexts, yielding circular traffic under this approach. The second solution is last resort. For conjunctions it would amount to a huge number of ambiguities, none of which would be locally decidable. But what is more: languages that have propositional conjunction, will use this conjunction for all other envirinments. If conjunctions are multiple ambiguous, one would expect languages to come up with specialized elements. Though some

language have a specilaized element for group formation (a kind of 'with' element), (see hoe-heet-ie-ook-alweer-met-dat=boek-over-koordinatie) these languages don't have sentential coordination at all.
The propositional (and boolean) base of coordination is reflected in the proposal in Rooth and Partee (1983) to interpret all coordinations as the meet on propositions: $|[|[a]| \text{ and } |[b]|]| = \lambda\varphi.\ [\varphi(|[a]|) \wedge \varphi(|[b]|)]$, where φ's type can be derived from the coordinates' type. The argument to this function must be provided by the sentential construction surrounding the coordination. Here the local indeterminism of coordination strikes back: there is no such remnant environment without parsing the whole construction and determining the coordinates. Though Rooth and Partee succeed in generalizing the typological aspect of the meaning of the coordination, their proposal does not solve the conjunction paradox: in order to determine what a coordination contributes to the semantic construal of parsing, parsing must be completed including the determination of the coordinates. It is because of this paradox that I claim that there is no semantic construal for coordination. The meaning of *John and Bill* and *John to swim and Bill to walk* does not exist independently of the semantic construal of a sentence. In this sense, coordinations are anaphoric.

Ades … and Steedman, M., ….*in Libguistics and philosophy 1980*
Barbiers, S., *dissertatie*, 1995
Benthem, J. van, *Essais in logical semantics*, Kluwer , 1986
Benthem, J. van, *Language in Action*, North Holland, 1991
Bouma, G. and. G. van Noord, `Constraint-based Categorial Grammar', *Proceedings 32nd Annual Meeting of the ACL*, ACL 1994, p. 147 - 154
Buszkowski, W., 'Compatability of a categorial grammar with an associated category system', *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 28, p. 229-237, 1982
Carpenter, B., *Type-Logical Semantics*, MIT Press, 1997
Chomsky, N., *The Minimalist program*, MIT Press ,1995
Cormack, A., and N. Smith, 'Where is a sign merged?', *Glot International 4:6*, 1999, p. 20
Cremers, C. and M. Hijzelendoorn, 'Filtering Left Dislocation Chains in Parsing Categorial Grammar', *in*: J. Landsbergen et al. (eds), *Computational Linguistics in the Netherlands 1996*, IPO TU Eindhoven, 1997b
Cremers, C. and M. Hijzelendoorn, 'Pruning Search Space for Parsing Free Coordination in Categorial Grammar', in: *International Workshop on Parsing Technologies. Proceedings 1997*, MIT, 1997a
Cremers, C., *On Parsing Coordination Categorially*, HIL dissertations 5, Leiden University, 1993
Cremers, C., (On disjarmony en zo), EACL Proceedings, 1999
Den Besten, H. …. tekst over derde konstruktie
Flynn, M., (diss), 1983
Friedman, J., D. Dai, W. Wang, 'The weak generative capacity of parenthesis-free categorial grammars', *Proceedings of Coling 86*, ACL 1986, pp. 199 - 201
Hendriks, H., *Studied Flexibility. Categories and Types in Syntax and Semantics*, ILLC Dissertation Series 1993-5, University of Amsterdam, 1993
Hepple, M., (diss), 1990
Hepple, M., 'A Compilation-Chart Method for Linear Categorial Deduction', *Proceedings Coling 1996*, Center for Sprogteknologi, Copenhagen, 1996
Hoeksema, J. and R. Janda, '…', in: D. Oerhle, E. Bach and D. Wheeler (eds), *Categorial Grammar and natural Language*, Kluwer, 1988
Icke, V., *...The symmetry of the universe…*, 1996
Jacobson, P., `Comment Flexible Categorial Grammars', *in*: R. Levine (ed), *Formal grammar: theory and implementation*, Oxford UnivPress, 1991, p. 129 - 167
Janssen, T.M.V., Foundations and applications of Montague grammatica??, diss, UVA, 1986??
janssen, T.M.V., ''Compositionality', in: J. van Benthem and A. ter Meulen (eds), *Handbook of Logic and Language*, Elsevier 1997, p. 417-474
Johannesen, …, diss, 1993
Joshi, A.K., K. Vijay-Shanker, D. Weir, `The Convergence of Mildly Context-Sensitive Grammar Formalisms', *in*: P. Sells, S.M. Shieber, T. Wasow (eds), *Foundational Issues in Natural Language Processing*, MIT Press, 1991, pp. 31 - 82
Kayne, R., *The antisymmetry of syntax*, MIT Press, 1994
König, E., *Der Lambek-Kalkül. Eine Logik für lexikalische Grammatiken*, Ph.diss University of Stuttgart, IBM IWBS Report 146, 1990
Moortgat, M., *Categorial Investigations*, Foris, 1988
Moortgat, M., '…….', in: J. van Benthem and A. ter Meulen (eds), *Handbook of Logic and Language*, 1998
Morrill, G., 'Higher-order Lineair Logic Programming of Categorial Deduction', *Proceedings of the Seventh Conference of the EACL*, University College Dublin, 1995, p. 133-140
Morrill, G., *Type Logical Grammar. Categorial Logic of Signs*, Kluwer, 1994

Link, G., ….. aricle on groups, ???1984

Partee, B. and M. Rooth, 'Generalized Conjunction and Type Ambiguity', *in*: R. Bäuerle, C. Schwarze and A. von Stechow (eds*), Meaning, Use and Interpretation in Language, De Gruyter*, 1983

Pollard, C., *Generalized Phrase Structure Grammar, Head Grammars and natural Languages*, PhD. diss., Stanford, 1984

Postma, GJ, diss, 1995

Retoré, C., and E. Stabler (eds), (Proceedings workshop on minimalism and multimodal categorial grammar), ESLSI 1999

Roorda, D., *Resource logics: prooftheoretical investigations*, Ph.diss., University of Amsterdam, 1991

Stabler, E., *….and Language Acquisition*, 1996

Steedman, M., 'Gapping as Constituent Coordination', *Linguistics and Philosophy 13:2*, 1990, p. 207-264

Steedman, M., *Surface Structure and Interpretation*, MIT Press, Cambridge, 1996

Van Benthem, J., *Essays in Logical Semantics*, Reidel, 1986

Vermaat, W., *Controlling Movement: Minimalism in a dedective perspective*, Ms thesis, CKI Doctoraal Scripties, Utrecht University, 1999

Versmissen, K., *Grammatical composition: modes, models, modalities. Logical and inguistic aspects of multimodal categorial grammars*, PhD. diss, OTS, Utrecht university, 1996

Zwart, C.J.W., *Dutch Syntax. A minimalist approach*, Ph. diss, University of Groningen, 1993

Zwarts, F., *Categorial grammatica en algenraïsche semantiek*, diss Rijksuniversiteit Groningen, 1986