On Calculus of Displacement*

Glyn Morrill
Departament de LSI
Universitat Politècnica de Catalunya

Oriol Valentín
Barcelona Media, Centre d'Innovació
Universitat Pompeu Fabra

1 Introduction

The calculus of Lambek (1958) did not make much impact until the 1980s, but for more than twenty years now it has constituted the foundation of type logical categorial grammar. It has occupied such a central position because of its good logical properties, but it has also been clear, even from the start of its renaissance, that the Lambek calculus suffers from fundamental shortcomings, which we shall mention below.

Certainly it seems that the way ahead in logical categorial grammar is to enrich the Lambek calculus with additional connectives. Thus it was proposed to add intersection and union as far back as Lambek (1961), and such extension is what is meant by type logical categorial grammar. Particular inspiration came from linear logic. Propositional linear connectives divide into additives, multiplicatives, and exponentials. Technically, the Lambek connectives are (noncommutative) linear multiplicatives, so it is natural to consider enrichment of Lambek calculus with (noncommutative) additives and exponentials as well. Quantifiers may also be added (Morrill, 1994, ch. 6) and unary modalities (Morrill 1990, 1992; Moortgat 1995).

However, none of these extensions address the essential limitation of the Lambek basis, which is as follows. The Lambek calculus is a sequence logic of concatenation. This is all well and good in that words are arranged sequentially, however natural language exhibits action at a distance: dependencies which are discontinuous. The Lambek calculus can capture some discontinuous dependencies, namely those in which the discontinuous dependency is peripheral. But it cannot capture the same kinds of dependencies when they are nonperipheral, i.e. medial. In this respect the foundation provided by the Lambek calculus is fundamentally imperfect.

A major proposal to refound categorial grammar was made in Moortgat (1997). Observing that the binary Lambek connectives form residuated triples, and unary modalities residuated pairs, Moortgat proposed to multiply such connective families, defining each family with respect to a different primitive mode of composition represented in the metalinguistic sequent punctuation in the inference rules for the connectives of the family. In such multimodal type logical grammar the modes are interrelated by structural rules defining equations and inclusions on the sequent configurations formed by their composition. This constitutes a powerful methodology and has been the inspiration of type logical grammar for a generation, but the addition of structural rules makes derivation laborious to hand and eye, and creates a problematic search space computationally. No single calculus developed according to these design principles particularly stands out for the breadth and elegance of its empirical application. Indeed, it might be remarked that the motto of substructural logic is to drop structural rules, not to introduce more of them.

It is in this context that Morrill and Valentín (2010) offers the displacement calculus. Sure enough this multiplies residuated triple connective families, as in multimodal type logical grammar, but it has a unique primitive mode of composition, concatenation, like the Lambek calculus, and it has a unimodal sequent calculus, like the Lambek calculus. Thus, importantly, it is entirely free of structural rules, and apparently preserves all the other good

*The research reported in the present paper was supported by DGICYT project SESAAME-BAR (TIN2008-06582-C03-01).

proof-theoretic properties of the Lambek calculus. Linguistically, and as illustrated in Morrill and Valentín (2010), it is the generalization of the Lambek calculus which has the widest and most economical coverage that we are aware of. The present paper enters into technical consideration of the displacement calculus given this state of affairs. We define this generalization of the Lambek calculus and consider some of the non-context free properties it characterizes.

2 Displacement calculus

The Lambek calculus (Lambek 1958) forms the basis of type logical categorial grammar (Morrill 1994, Moortgat 1997, Morrill forthcoming). It is a sequence logic without structural rules which enjoys Cut-elimination, the subformula property, and decidability. It is intuitionistic, and so supports the standard Curry-Howard type-logical categorial semantics. In this connection it has the finite reading property. But as a logic of concatenation, the Lambek calculus can only analyse displacement when the dependencies happen to be peripheral. As a consequence it cannot account for the syntax and semantics of:

(1)
- Discontinuous idioms (*Mary gave the man the cold shoulder*).
- Quantification (*John gave every book to Mary*; *Mary thinks someone left*; *Everyone loves someone*).
- VP ellipsis (*John slept before Mary did*; *John slept and Mary did too*).
- Medial extraction (*dog that Mary saw today*).
- Pied-piping (*mountain the painting of which by Cezanne John sold for $10,000,000.*
- Appositive relativization (*John, who jogs, sneezed*).
- Parentheticals (*Fortunately, John has perseverance*; *John, fortunately, has perseverance*; *John has, fortunately, perseverance*; *John has perseverance, fortunately*).
- Gapping (*John studies logic, and Charles, phonetics*).
- Comparative subdeletion (*John ate more donuts than Mary bought bagels*).
- Reflexivization (*John sent himself flowers*).

Furthermore, the Lambek calculus is context-free in generative power (Pentus 1992) and so cannot generate cross-serial dependencies as in Dutch and Swiss-German (Sheiber 1985).

The calculus of displacement, like the Lambek calculus, is a sequence logic without structural rules which enjoys Cut-elimination, the subformula property, and decidability (Morrill and Valentín 2010). Moreover, like the Lambek calculus it is intuitionistic, and so supports the standard categorial Curry-Howard type-logical semantics. In this relation it has the finite reading property. It is a logic not only of concatenation but also of intercalation and provides basic analyses of all of the phenomena itemized in (1) (Morrill and Valentín 2010). Furthermore it analyses verb raising and cross-serial dependencies (Morrill, Valentín and Fadda 2009).

The types of the calculus of displacement **D** classify strings over a vocabulary including a distinguished placeholder 1 called the *separator*. The sort $i \in \mathcal{N}$ of a (discontinuous) string is the number of separators it contains and these punctuate it into $i+1$ continuous substrings. The types of **D** are sorted into types $\mathcal{F}_i$ of sort i as follows:

(2)

$\mathcal{F}_j$	:=	$\mathcal{F}_i \backslash \mathcal{F}_{i+j}$	under
$\mathcal{F}_i$	:=	$\mathcal{F}_{i+j} / \mathcal{F}_j$	over
$\mathcal{F}_{i+j}$	:=	$\mathcal{F}_i \cdot \mathcal{F}_j$	product
$\mathcal{F}_0$	:=	I	product unit
$\mathcal{F}_j$	:=	$\mathcal{F}_{i+1}\downarrow_k \mathcal{F}_{i+j}, 1 \leq k \leq i+1$	infix
$\mathcal{F}_{i+1}$	:=	$\mathcal{F}_{i+j}\uparrow_k \mathcal{F}_j, 1 \leq k \leq i+1$	extract
$\mathcal{F}_{i+j}$	:=	$\mathcal{F}_{i+1} \odot_k \mathcal{F}_j, 1 \leq k \leq i+1$	disc. product
$\mathcal{F}_1$	:=	J	disc. prod. unit

Where A is a type we call its sort sA. We present the calculus using a special kind of sequent calculus which we call hypersequent calculus. The set $\mathcal{O}$ of *hyperconfigurations* is defined as follows, where Λ is the empty string and $[\,]$ is the metalinguistic separator:

$$(3)\quad \mathcal{O} ::= \Lambda \mid [\,] \mid \mathcal{F}_0 \mid \mathcal{F}_{i+1}\{\underbrace{\mathcal{O} : \ldots : \mathcal{O}}_{i+1\ \mathcal{O}'s}\} \mid \mathcal{O}, \mathcal{O}$$

Note that the hyperconfigurations are of a new kind in which some type formulas, namely the type formulas of sort greater than one, label mother nodes rather than leaves, and have a number of immediate subhyperconfigurations equal to their sort. This signifies a discontinuous type intercalated by these subhyperconfigurations. Thus $A\{\Delta_1 : \ldots : \Delta_n\}$ interpreted syntactically is formed by strings $\alpha_0 + \beta_1 + \alpha_1 + \cdots + \alpha_{n-1} + \beta_n + \alpha_n$ where $\alpha_0 + 1 + \alpha_1 + \cdots + \alpha_{n-1} + 1 + \alpha_n \in A$ and $\beta_1 \in$

$\Delta_1, \ldots, \beta_n \in \Delta_n$. We call these types *hyperleaves* since in multimodal calculus they would be leaves. The sort of a hyperconfiguration is the number of separators it contains. A *hypersequent* $\Gamma \Rightarrow A$ comprises an antecedent hyperconfiguration Γ of sort i and a succedent type A of sort i. The *vector* $\overrightarrow{A}$ of a type A is defined by:

$$(4)\quad \overrightarrow{A} = \begin{cases} A & \text{if } sA = 0 \\ A\{\underbrace{[\,] : \ldots : [\,]}_{sA\ [\,]'s}\} & \text{if } sA > 0 \end{cases}$$

Where Γ_1 is a hyperconfiguration of sort at least k and Γ_2 is a hyperconfiguration, $\Gamma_1|_k\Gamma_2$ signifies the hyperconfiguration which is the result of replacing by Γ_2 the kth separator in Γ_1. Where Γ is a hyperconfiguration of sort i and $\Phi_1, \ldots, \Phi_i$ are hyperconfigurations, the generalized wrap $\Gamma \otimes \langle \Phi_1, \ldots, \Phi_i \rangle$ is the result of simultaneously replacing the successive separators in Γ by $\Phi_1, \ldots, \Phi_i$ respectively. In the hypersequent calculus the discontinuous distinguished hyperoccurrence notation $\Delta\langle\Gamma\rangle$ refers to a hyperconfiguration Δ and continuous subhyperconfigurations $\Delta_1, \ldots, \Delta_i$ and a discontinuous subhyperconfiguration Γ of sort i such that $\Gamma \otimes \langle \Delta_1, \ldots, \Delta_i \rangle$ is a continuous subhyperconfiguration. Technically, whereas the usual distinguished occurrence notation $\Delta(\Gamma)$ refers to a context containing a *hole* which is a leaf, in hypersequent calculus the distinguished hyperoccurrence notation $\Delta\langle\Gamma\rangle$ refers to a context containing a hole which may be a hyperleaf, a *hyperhole*. The hypersequent calculus for the calculus of displacement is given in Figure 1.

3 Displacement grammars

We now turn to **D**-grammars and the languages they generate.

Given a vocabulary $V = \Sigma \cup \{1\}$ a lexical assignment α: A comprises a type A and a string $\alpha \in V^+ - \{\underbrace{1 + \cdots + 1}_{n} : n > 0\}$ of sort sA. A lexicon is a finite set of lexical assignments.

We define a *labelling* σ of a hyperconfiguration Δ as a mapping sending each type occurrence A in Δ to a string of sort sA. A *labelled hyperconfiguration* Δ^σ comprises a hyperconfiguration Δ and a labelling σ of Δ. We define the *yield* of a labelled hyperconfiguration Δ^σ as follows:

$$\frac{}{\overrightarrow{A} \Rightarrow A}\,id \qquad \frac{\Gamma \Rightarrow A \qquad \Delta\langle\overrightarrow{A}\rangle \Rightarrow B}{\Delta\langle\Gamma\rangle \Rightarrow B}\,Cut$$

$$\frac{\Gamma \Rightarrow A \qquad \Delta\langle\overrightarrow{C}\rangle \Rightarrow D}{\Delta\langle\Gamma, \overrightarrow{A\backslash C}\rangle \Rightarrow D}\backslash L \qquad \frac{\overrightarrow{A}, \Gamma \Rightarrow C}{\Gamma \Rightarrow A\backslash C}\backslash R$$

$$\frac{\Gamma \Rightarrow B \qquad \Delta\langle\overrightarrow{C}\rangle \Rightarrow D}{\Delta\langle\overrightarrow{C/B}, \Gamma\rangle \Rightarrow D}/L \qquad \frac{\Gamma, \overrightarrow{B} \Rightarrow C}{\Gamma \Rightarrow C/B}/R$$

$$\frac{\Delta\langle\overrightarrow{A}, \overrightarrow{B}\rangle \Rightarrow D}{\Delta\langle\overrightarrow{A \cdot B}\rangle \Rightarrow D}\cdot L \qquad \frac{\Gamma_1 \Rightarrow A \qquad \Gamma_2 \Rightarrow B}{\Gamma_1, \Gamma_2 \Rightarrow A \cdot B}\cdot R$$

$$\frac{\Delta\langle\Lambda\rangle \Rightarrow A}{\Delta\langle\overrightarrow{I}\rangle \Rightarrow A}\,IL \qquad \frac{}{\Lambda \Rightarrow I}\,IR$$

$$\frac{\Gamma \Rightarrow A \qquad \Delta\langle\overrightarrow{C}\rangle \Rightarrow D}{\Delta\langle\Gamma|_k \overrightarrow{A\downarrow_k C}\rangle \Rightarrow D}\downarrow_k L \qquad \frac{\overrightarrow{A}|_k\Gamma \Rightarrow C}{\Gamma \Rightarrow A\downarrow_k C}\downarrow_k R$$

$$\frac{\Gamma \Rightarrow B \qquad \Delta\langle\overrightarrow{C}\rangle \Rightarrow D}{\Delta\langle\overrightarrow{C\uparrow_k B}|_k\Gamma\rangle \Rightarrow D}\uparrow_k L \qquad \frac{\Gamma|_k\overrightarrow{B} \Rightarrow C}{\Gamma \Rightarrow C\uparrow_k B}\uparrow_k R$$

$$\frac{\Delta\langle\overrightarrow{A}|_k\overrightarrow{B}\rangle \Rightarrow D}{\Delta\langle\overrightarrow{A\odot_k B}\rangle \Rightarrow D}\odot_k L \qquad \frac{\Gamma_1 \Rightarrow A \qquad \Gamma_2 \Rightarrow B}{\Gamma_1|_k\Gamma_2 \Rightarrow A\odot_k B}\odot_k R$$

$$\frac{\Delta\langle[\,]\rangle \Rightarrow A}{\Delta\langle\overrightarrow{J}\rangle \Rightarrow A}\,JL \qquad \frac{}{[\,] \Rightarrow J}\,JR$$

Figure 1: Calculus of displacement **D**

$$(5)\quad \begin{array}{l} yield(\Lambda^\sigma) = \Lambda \\ yield([\,]^\sigma) = 1 \\ yield((\Delta, \Gamma)^\sigma) = yield(\Delta^\sigma) + yield(\Gamma^\sigma) \\ yield(A^\sigma) = \sigma(A) \text{ for } A \text{ of sort} 0 \\ yield((A\{\Delta_1 : \cdots : \Delta_{sA}\})^\sigma) = \\ a_1 + yield(\Delta_1^\sigma) + a_2 + yield(\Delta_2^\sigma) + \cdots + \\ a_{sA-1} + yield(\Delta_{sA}^\sigma) + a_{sA} \end{array}$$

where in the last line of the definition A is of sort greater than 0 and $\sigma(A)$ is $a_1 + 1 + a_2 + \cdots + a_{sA-1} + 1 + a_{sA}$.

A labelling σ of a hyperconfiguation Δ is *compatible* with a lexicon **Lex** if and only if $\sigma(A)$: $A \in$ **Lex** for every A in Δ. The language $L($**Lex**$, A)$ generated from lexicon **Lex** for type A is defined as follows:

(6) $L($**Lex**$, A) = \{yield(\Delta^\sigma) |$ such that $\Delta \Rightarrow A$ is a theorem of **D** and σ is compatible with **Lex**$\}$

Theorem 1 *The problem of recognition in the class of* **D***-grammars is decidable.*

Proof. Since for every labelling σ compatible with a lexicon for every type A, $\sigma(A)$ contains at least one symbol different from 1, the set of labelled hyperconfigurations such that their yield equals a given α is finite. Now as theoremhood in the **D** is decidable we have then that

the problem of recognition is decidable since it reduces to a finite number of tests of theoremhood. □

A Prolog parser/theorem-prover for the calculus of displacement has been implemented. It operates by Cut-free backward-chaining hypersequent proof search.

4 Some non-context free D-languages

Computer-generated output for the lexicon and analyses of Dutch verb raising and cross-serial dependencies are given in Appendix A. (We abbreviate $\downarrow_1$, $\odot_1$ and $\uparrow_1$ as $\downarrow$, $\odot$ and $\uparrow$; only discontinuities with a single separator are considered in this paper.)

The non-context free language $\{a^n b^n c^n |\ n > 0\}$ is generated by the following assignments where $sA = sB = sC = 1$ and the distinguished type is $A \odot I$.

(7) b: $J \backslash B, J \backslash (A \downarrow B)$
c: $B \backslash C$
a: A / C

The assignment b: $J \backslash B$ generates $1{+}b$: B. Then combination with the assignment for c generates $1{+}b{+}c$: C and combination of this with the assignment for a gives $a{+}1{+}b{+}c$: A. Wrapping this around the product unit gives $a{+}b{+}c$: $A \odot I$; alternatively b: $J \backslash (A \downarrow B)$ which gives $1{+}b$: $A \downarrow B$ can infix to form $a{+}1{+}b{+}b{+}c$: B which combines with c and a again, and so on.

The non-context free copy language $\{ww |\ w \in \{a, b\}^+\}$ is generated by the following assignments where $sA = sB = 0$ and $sS = 1$ and the distinguished type is $S \odot I$.

(8) a: $J \backslash (A \backslash S), J \backslash (S \downarrow (A \backslash S)), A$
b: $J \backslash (B \backslash S), J \backslash (S \downarrow (B \backslash S)), B$

Let G be a rewrite grammar containing productions of the form $A \to a$ and $B \to cD \mid Dc$. Replacing the former by a: A and the latter by c: $(D \uparrow I) \downarrow B$ gives a displacement grammar which generates the permutation closure of $L(G)$. It follows that there is a displacement grammar for every language Mix_n of strings with equal numbers of symbols $a_1, \ldots, a_n$. In particular, the non context-free language $Mix = \{w \in \{a, b, c\} |\ |w|_a = |w|_b = |w|_c > 0\}$ is generated by the following assignments:

(9) a: $a, (S \uparrow I) \downarrow a$
b: $(a \uparrow I) \downarrow b$
c: $(b \uparrow I) \downarrow S$

Here $sA = sB = sC = 0$ and the distinguished type is S. Appendix B contains a sample derivation of this displacement grammar for Mix.

5 A lower bound on the recognizing power of D-grammars

In this section we prove that **D**-grammars recognize the permutation closures of context-free languages.

This result is obtained using a restricted fragment of the calculus. We define the set $T = \{A |\ A \text{ is an atomic type}\} \cup \{(A \uparrow I) \downarrow B |\ A \text{ and } B \text{ are atomic types}\}$. A *T-hypersequent* is a hypersequent such that the types of the antecedent belong to T and the succedent is an atomic type.

Lemma 2 (Rearrangement lemma) *Let $\Delta \Rightarrow S$ be a provable T-hypersequent. Then, where $\mathcal{D}$ is a derivation of $\Delta \Rightarrow S$, $\mathcal{D}$ can be rearranged into a new derivation $\mathcal{D}^\star$ of $\Delta \Rightarrow S$ in such a way that the height of $\mathcal{D}$ is preserved, and the last rule of $\mathcal{D}^\star$ has an axiom $S \Rightarrow S$ as the right premise, i.e.:*

$$\mathcal{D} \vdash \dfrac{\vdots}{\Delta \Rightarrow S} \downarrow L \quad \rightsquigarrow \quad \mathcal{D}^\star \vdash \dfrac{\Gamma([\,]) \Rightarrow \breve{}A \qquad S \Rightarrow S}{\Delta \Rightarrow S} \downarrow L$$

where $\Delta = \Gamma(\breve{}A \downarrow S)$ for some atomic type A.

Proof.

$$\dfrac{\Gamma([\,]) \Rightarrow \breve{}P \qquad \dfrac{\Delta(Q; [\,]) \Rightarrow \breve{}R \qquad S \Rightarrow S}{\Delta(Q; \breve{}R \downarrow S) \Rightarrow S} \downarrow L}{\Delta(\Gamma(\breve{}P \downarrow Q); \breve{}R \downarrow S) \Rightarrow S} \downarrow L$$

$$\rightsquigarrow$$

$$\dfrac{\dfrac{\Gamma([\,]) \Rightarrow \breve{}P \qquad \Delta(Q; [\,]) \Rightarrow \breve{}R}{\Delta(\Gamma(\breve{}P \downarrow Q); [\,]) \Rightarrow \breve{}R} \downarrow L \qquad S \Rightarrow S}{\Delta(\Gamma(\breve{}P \downarrow Q); \breve{}R \downarrow S) \Rightarrow S} \downarrow L$$

□

Lemma 3 (Fronting lemma) *Let $\Delta(A) \Rightarrow S$ be a provable T-hypersequent with a distinguished occurrence of type A. Then:*

$$\vdash A, \Delta(\Lambda) \Rightarrow S$$

Proof. We proceed by induction on the length of hypersequents. We shall write $A_1, \cdots, A_j, \cdots, A_n \Rightarrow S$ for $\Delta(A)$ where we consider A_j as the distinguished occurrence we want to be displaced to the left of the antecedent. By the previous lemma, $A_1, \cdots, A_j, \cdots, A_n \Rightarrow S$ has a derivation with last rule:[1]

$$\frac{A_1, \cdots, A_{i-1}, [\,], A_{i+1} \cdots, A_j, \cdots, A_n \Rightarrow R{\uparrow}I \qquad S \Rightarrow S}{A_1, \cdots, A_{i-1}, (R{\uparrow}I){\downarrow}S, A_{i+1} \cdots, A_j, \cdots, A_n \Rightarrow S}{\downarrow}L$$

Two cases are considered:

• Case $A_j \neq (R{\uparrow}I){\downarrow}S$:

We have $(R{\uparrow}I){\odot}I \Rightarrow R$ is provable. By applying the Cut rule to the left premise of the last rule, we derive:

$$A_1, \cdots, A_{i-1}, \Lambda, A_{i+1} \cdots, A_j, \cdots, A_n \Rightarrow R$$

Hence by induction hypothesis:

$$\vdash A_j, A_1, \cdots, A_{i-1}, \Lambda, A_{i+1} \cdots, \Lambda, \cdots, A_n \Rightarrow R$$

We apply now the $\uparrow$ right rule after the introduction of the unit I:

$$\frac{A_j, A_1, \cdots, A_{i-1}, I, A_{i+1} \cdots, \Lambda, \cdots, A_n \Rightarrow R}{A_j, A_1, \cdots, A_{i-1}, [\,], A_{i+1} \cdots, \Lambda, \cdots, A_n \Rightarrow R{\uparrow}I}{\uparrow}R$$

By the $\downarrow$ left rule:

$$\frac{A_j, A_1, \cdots, A_{i-1}, [\,], A_{i+1} \cdots, \Lambda, \cdots, A_n \Rightarrow R{\uparrow}I \qquad S \Rightarrow S}{A_j, A_1, \cdots, A_{i-1}, (R{\uparrow}I){\downarrow}S, A_{i+1} \cdots, \Lambda, \cdots, A_n \Rightarrow S{\uparrow}I}{\downarrow}L$$

In this case, we have proved the fronting lemma.

• Case $A_j = (R{\uparrow}I){\downarrow}S$:

As before we have the following provable hypersequent:

$$A_1, \cdots, A_{i-1}, \Lambda, A_{i+1}, \cdots, A_n \Rightarrow R$$

By the right $\uparrow$ rule after the introduction of I, we derive:

$$[\,], A_1, \cdots, A_{i-1}, \Lambda, A_{i+1} \cdots, A_j, \cdots, A_n \Rightarrow R{\uparrow}I$$

By the left $\downarrow$ rule:

$$\frac{[\,], A_1, \cdots, A_{i-1}, \Lambda, A_{i+1}, \cdots, A_n \Rightarrow R{\uparrow}I \qquad S \Rightarrow S}{(R{\uparrow}I){\downarrow}S, A_1, \cdots, A_{i-1}, \Lambda, A_{i+1}, \cdots, A_n \Rightarrow S}{\downarrow}L$$

We have proved the fronting lemma in case $A_j = (R{\uparrow}I){\downarrow}S$. In both cases then, the lemma is proved. □

[1] Without loss of generality we write A_j to the right of $(R{\uparrow}I){\downarrow}S$.

We now show how the permutation closure of any regular language (excluding the empty string) can be recognized by a **D**-grammar. Let $G = (N, \Sigma, P, S)$ be a regular grammar. Suppose G is right-linear. We define a **D**-grammar comprising a lexicon $\mathbf{Lex_G}$ with atomic types the nonterminals N of G. The vocabulary of $\mathbf{Lex_G}$ is $\Sigma \cup \{1\}$. For every production of the form $A \to c$ with A nonterminal and $c \in \Sigma$, we stipulate that c: $A \in \mathbf{Lex_G}$. And for every production of the form $B \to cA$ (with $A, B \in N$ and $c \in \Sigma$), we stipulate c: $(A{\uparrow}I){\downarrow}B \in \mathbf{Lex_G}$. We want to prove that the language recognized by $\mathbf{Lex_G}$ with distinguished symbol S is the permutation closure of the language generated by G: $L(\mathbf{Lex_G}, S) = Perm(L(G, S))$. The following lemmas prove the equation.

Lemma 4 $L(G, S) \subseteq L(\mathbf{Lex_G}, S)$.

The proof of this lemma proceeds by a simple induction on the length of the derivations of G. The base case is obvious. For the inductive case, suppose we have the derivation whose rewritten string is $a_1 \cdots a_n A$ such that $A \to cB \in P$. Then by induction hypothesis $a_1 + \cdots + a_n \in L(G, B) \subseteq L(\mathbf{Lex_G}, B)$. Hence there exists a labeled hyperconfiguration Δ^σ whose types belong to the types of $\mathbf{L_G}$, $\vdash \Delta \Rightarrow B$ and the yield of Δ^σ is $a_1 + \cdots + a_n$; after the introduction of the unit:

$$\frac{\dfrac{I, \Delta \Rightarrow B}{[\,], \Delta \Rightarrow B{\uparrow}I}{\uparrow}R \qquad A \Rightarrow A}{(B{\uparrow}I){\downarrow}A, \Delta \Rightarrow A}{\downarrow}L$$

Now, c: $(B{\uparrow}I){\downarrow}A \in \mathbf{Lex_G}$. Hence $c + a_1 + \cdots + a_n \in L(\mathbf{Lex_G}, \mathbf{A})$.

Lemma 5 $Perm(L(G, S)) \subseteq L(\mathbf{Lex_G}, S)$

Proof. Let $\Delta \Rightarrow S$ be a provable hypersequent with a compatible labelling such that the yield of Δ is $w \in L(G, S)$ and the types occurring in Δ belong to the set of types of $\mathbf{Lex_G}$:

$$a_1 : A_1, \cdots, a_n : A_n \Rightarrow S,\ w = a_1 + \cdots + a_n$$

By the fronting lemma, any type A_i can be fronted, i.e.: $\vdash a_i : A_i, a_1 : A_1, \cdots, a_{i-1} : A_{i-1}, \Lambda, a_{i+1} : A_{i+1}, \cdots, a_n : A_n \Rightarrow S$. By repeating this process via the fronting lemma, any permutation of the initial w can be obtained. □

Lemma 6 $L(\mathbf{Lex_G}, S) \subseteq Perm(L(G, S))$

Proof. We prove that for every atomic type $A \in N$, $L(\mathbf{Lex_G}, A) \subseteq Perm(L(G, A))$. This entails in particular $L(\mathbf{Lex_G}, S) \subseteq Perm(L(G, S))$ where $S \in N$ is the distinguished nonterminal symbol. The proof goes by induction on the height of derivations of hypersequents $\Delta \Rightarrow A$ such that the types of Δ belong to the types of $\mathbf{Lex_G}$. Case that the height is 0: let $A \in N$ be such that $A \to a$ is a production of G and $a \in V$. Then $a : A \in \mathbf{Lex}$. Case that the height is greater than 0: suppose we have a $\mathbf{Lex_G}$-derivation of $\Delta \Rightarrow A$ with height $n+1$. By the rearrangement lemma, the derivation of $\Delta \Rightarrow A$ can modified in such a way that the height of the derivation is preserved and one of the premises is an axiom:

$$\dfrac{\Delta([\,]) \Rightarrow A{\uparrow}I \qquad S \Rightarrow S}{\Delta((A{\uparrow}I){\downarrow}S) \Rightarrow S}\,{\downarrow}L$$

By a simple reasoning we have $\Delta(\Lambda) \Rightarrow A$ with height lesser or equal than the height of $\Delta([\,]) \Rightarrow A{\uparrow}I$. Since the types of $\Delta(\Lambda)$ belong to the types of $\mathbf{Lex_G}$ we can apply the induction hypothesis, and then we have that $L(\mathbf{Lex_G}, A) \subseteq Perm(L(G, A))$. Now, every $w \in L(\mathbf{Lex_G}, A)$ is the permutation of some $\widetilde{w} \in L(G, A)$. If we apply the rule $S \to cA$ we get $c + \widetilde{w} \in L(G, S)$. But c: $(A{\uparrow}I){\downarrow}S$. Hence, if we insert c in w we get a permutation of $c + \widetilde{w}$. □

Theorem 7 *For every regular grammar G we have $L(\mathbf{Lex_G}, S) = Perm(L(G, S))$*

Corollary 8 *For every context-free language L, the permutation closure of L $Perm(L)$ is recognized by a* **D***-grammar.*

Proof. By an argument invoking properties of semi-linear sets,[2] we know that any permutation closure of a context-free language is equal to the permutation closure of some regular language. This reduces the proof of this corollary to the class of regular languages. The previous theorem proves it. □

[2]See van Benthem (1991).

Appendix A. Computer-generated output for Dutch verb raising and cross-serial dependencies

boeken : N : *books*
cecilia : N : c
de : N/CN : ι
jan : N : j
helpen : $J\backslash((N\backslash Si){\downarrow}(N\backslash(N\backslash Si))) : \lambda A\lambda B\lambda C\lambda D((help\ D)\ (B\ C))$
henk : N : h
kan : $(N\backslash Si){\downarrow}(N\backslash S) : \lambda A\lambda B((isable\ B)\ (A\ B))$
kunnen : $J\backslash((N\backslash Si){\downarrow}(N\backslash Si)) : \lambda A\lambda B\lambda C((beable\ C)\ (B\ C))$
las : $N\backslash(N\backslash S) : reads$
lezen : $J\backslash(N\backslash(N\backslash Si)) : \lambda Aread$
nijlpaarden : CN : *hippos*
voeren : $J\backslash(N\backslash(N\backslash Si)) : \lambda Afeed$
wil : $(N\backslash Si){\downarrow}(N\backslash S) : \lambda A\lambda B((wants\ B)\ (A\ B))$
zag : $(N\backslash Si){\downarrow}(N\backslash(N\backslash S)) : \lambda A\lambda B\lambda C((saw\ C)\ (A\ B))$

(1) **jan+boeken+las** : S

$N : j, N : books, N\backslash(N\backslash S) : reads \ \Rightarrow\ S$

$$\dfrac{\dfrac{}{N \ \Rightarrow\ N} \qquad \dfrac{\dfrac{}{N \ \Rightarrow\ N} \quad \dfrac{}{S \ \Rightarrow\ S}\,\backslash L}{N, N\backslash S \ \Rightarrow\ S}\,\backslash L}{N, N, N\backslash(N\backslash S) \ \Rightarrow\ S}$$

$((reads\ books)\ j)$

(2) **jan+boeken+kan+lezen** : S

$N : j, N : books, (N\backslash Si){\downarrow}(N\backslash S) : \lambda A\lambda B((isable\ B)\ (A\ B)), J\backslash(N\backslash(N\backslash Si)) : \lambda Aread \ \Rightarrow\ S$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{N \Rightarrow N} \quad \dfrac{}{N \Rightarrow N}}{N, N\backslash S, \{[\,]\} \Rightarrow Si}\,\backslash L \quad \dfrac{}{Si \Rightarrow Si}}{N, N\backslash S, \{[\,]\} \Rightarrow Si}\,\backslash L \quad \dfrac{}{[\,] \Rightarrow J}\,JR}{N, N, [\,], J\backslash(N\backslash(N\backslash Si)) \Rightarrow Si}\,\backslash L}{N, [\,], J\backslash(N\backslash(N\backslash Si)) \Rightarrow N\backslash Si}\,\backslash R}{N, N, [\,], J\backslash(N\backslash(N\backslash Si)), (N\backslash(N\backslash Si))\backslash(N\backslash S) \Rightarrow S}\,{\downarrow}L \quad \dfrac{\dfrac{}{N \Rightarrow N} \quad \dfrac{}{S \Rightarrow S}}{N, N\backslash S \Rightarrow S}\,\backslash L}{S \Rightarrow ((N\backslash(N\backslash Si))\backslash(N\backslash(N\backslash Si)))\backslash(N\backslash S), N, N}\,{\uparrow}L$$

$((isable\ j)\ ((read\ books)\ j))$

(3) **jan+boeken+wil+kunnen+lezen** : S

$N : j, N : books, (N\backslash Si){\downarrow}(N\backslash S) : \lambda A\lambda B((wants\ B)\ (A\ B)), J\backslash((N\backslash Si){\downarrow}(N\backslash Si)) : \lambda A\lambda B\lambda C((beable\ C)\ (B\ C)), J\backslash(N\backslash(N\backslash Si)) : \lambda Aread \ \Rightarrow\ S$

$$\dfrac{\dfrac{}{[\,]\ \Rightarrow\ J}\,JR \quad \dfrac{\dfrac{}{N\ \Rightarrow\ N} \quad \dfrac{\dfrac{}{N\ \Rightarrow\ N} \quad \dfrac{}{Si\{[\,]\}\ \Rightarrow\ Si}\,\backslash L}{N, N\backslash Si\{[\,]\}\ \Rightarrow\ Si}\,\backslash L}{N, N, N\backslash(N\backslash Si)\{[\,]\}\ \Rightarrow\ Si}\,\backslash L}{N, N, [\,], J\backslash(N\backslash(N\backslash Si))\ \Rightarrow\ Si}\,\backslash L$$

$$\dfrac{\dfrac{}{[\,]\ \Rightarrow\ J}\,JR \quad \dfrac{N, N, [\,], J\backslash(N\backslash(N\backslash Si))\ \Rightarrow\ Si}{N, [\,], J\backslash(N\backslash(N\backslash Si))\ \Rightarrow\ N\backslash Si}\,\backslash R \quad \dfrac{\dfrac{}{N\ \Rightarrow\ N} \quad \dfrac{}{Si\{[\,]\}\ \Rightarrow\ Si}\,\backslash L}{N, N\backslash Si\{[\,]\}\ \Rightarrow\ Si}\,\backslash L}{N, N, (N\backslash Si)\downarrow(N\backslash Si)\{[\,]\}, J\backslash(N\backslash(N\backslash Si))\ \Rightarrow\ Si}\,\downarrow L}{N, N, (N\backslash Si)\downarrow(N\backslash Si), J\backslash((N\backslash Si)\downarrow(N\backslash Si)), J\backslash(N\backslash(N\backslash Si))\ \Rightarrow\ Si}\,\backslash L$$

$$\dfrac{N, N, [\,], J\backslash((N\backslash Si)\downarrow(N\backslash Si)), J\backslash(N\backslash(N\backslash Si))\ \Rightarrow\ N\backslash Si}{N, [\,], J\backslash((N\backslash Si)\downarrow(N\backslash Si)), J\backslash(N\backslash(N\backslash Si))\ \Rightarrow\ N\backslash Si}\,\backslash R \quad \dfrac{\dfrac{}{N\ \Rightarrow\ N} \quad \dfrac{}{S\ \Rightarrow\ S}\,\backslash L}{N, N\backslash S\ \Rightarrow\ S}\,\backslash L$$

$$\dfrac{N, N, [\,], J\backslash((N\backslash Si)\downarrow(N\backslash Si)), J\backslash(N\backslash(N\backslash Si))\ \Rightarrow\ N\backslash Si}{N, [\,], J\backslash((N\backslash Si)\downarrow(N\backslash Si)), J\backslash(N\backslash(N\backslash Si))\ \Rightarrow\ N\backslash Si} \quad \dfrac{N, N\backslash S\ \Rightarrow\ S}{N, N, (N\backslash Si)\downarrow(N\backslash S), J\backslash((N\backslash Si)\downarrow(N\backslash Si)), J\backslash(N\backslash(N\backslash Si))\ \Rightarrow\ S}\,\downarrow L$$

((*wants j*) ((*beable j*) ((*read books*) *j*)))

(4) **jan+cecilia+henk+de+nijlpaarden+zag+helpen+voeren** : S

$N : j, N : c, N : h, N/CN : \iota, CN : hippos, (N\backslash Si)\downarrow(N\backslash(N\backslash S)) : \lambda A\lambda B\lambda C((saw\ C)\ (A\ B)), J\backslash((N\backslash Si)\downarrow(N\backslash(N\backslash Si))) : \lambda A\lambda B\lambda C\lambda D((help\ D)\ (B\ C)), J\backslash(N\backslash(N\backslash Si)) : \lambda Afeed\ \Rightarrow\ S$

$$\dfrac{\dfrac{}{[\,]\ \Rightarrow\ J}\,JR \quad \dfrac{\dfrac{}{N\ \Rightarrow\ N} \quad \dfrac{\dfrac{}{N\ \Rightarrow\ N} \quad \dfrac{}{Si\{[\,]\}\ \Rightarrow\ Si}\,\backslash L}{N, N\backslash Si\{[\,]\}\ \Rightarrow\ Si}\,\backslash L}{N, N, N\backslash(N\backslash Si)\{[\,]\}\ \Rightarrow\ Si}\,\backslash L}{N, N, [\,], J\backslash(N\backslash(N\backslash Si))\ \Rightarrow\ Si}\,\backslash L$$

$$\dfrac{\dfrac{}{[\,]\ \Rightarrow\ J}\,JR \quad \dfrac{N, N, [\,], J\backslash(N\backslash(N\backslash Si))\ \Rightarrow\ Si}{N, [\,], J\backslash(N\backslash(N\backslash Si))\ \Rightarrow\ N\backslash Si}\,\backslash R \quad \dfrac{\dfrac{}{N\ \Rightarrow\ N} \quad \dfrac{\dfrac{}{N\ \Rightarrow\ N} \quad \dfrac{}{Si\{[\,]\}\ \Rightarrow\ Si}\,\backslash L}{N, N\backslash Si\{[\,]\}\ \Rightarrow\ Si}\,\backslash L}{N, N, N\backslash(N\backslash Si)\{[\,]\}\ \Rightarrow\ Si}\,\backslash L}{N, N, N, (N\backslash Si)\downarrow(N\backslash(N\backslash Si))\{[\,]\}, J\backslash(N\backslash(N\backslash Si))\ \Rightarrow\ Si}\,\downarrow L}{N, N, N, [\,], J\backslash((N\backslash Si)\downarrow(N\backslash(N\backslash Si))), J\backslash(N\backslash(N\backslash Si))\ \Rightarrow\ Si}\,\backslash L$$

$$\dfrac{\dfrac{}{CN\ \Rightarrow\ CN}}{N, N, N/CN, CN, [\,], J\backslash((N\backslash Si)\downarrow(N\backslash(N\backslash Si))), J\backslash(N\backslash(N\backslash Si))\ \Rightarrow\ Si}\,/L$$

$$\dfrac{N, N, N/CN, CN, [\,], J\backslash((N\backslash Si)\downarrow(N\backslash(N\backslash Si))), J\backslash(N\backslash(N\backslash Si))\ \Rightarrow\ Si}{N, N/CN, CN, [\,], J\backslash((N\backslash Si)\downarrow(N\backslash(N\backslash Si))), J\backslash(N\backslash(N\backslash Si))\ \Rightarrow\ N\backslash Si}\,\backslash R$$

$$\dfrac{\dfrac{}{N\ \Rightarrow\ N} \quad \dfrac{\dfrac{}{N\ \Rightarrow\ N} \quad \dfrac{}{S\ \Rightarrow\ S}\,\backslash L}{N, N\backslash S\ \Rightarrow\ S}\,\backslash L}{N, N, N\backslash(N\backslash S)\ \Rightarrow\ S}\,\backslash L$$

$$\dfrac{N, N/CN, CN, [\,], J\backslash((N\backslash Si)\downarrow(N\backslash(N\backslash Si))), J\backslash(N\backslash(N\backslash Si))\ \Rightarrow\ N\backslash Si \quad N, N, N\backslash(N\backslash S)\ \Rightarrow\ S}{N, N, N, N/CN, CN, (N\backslash Si)\downarrow(N\backslash(N\backslash S)), J\backslash((N\backslash Si)\downarrow(N\backslash(N\backslash Si))), J\backslash(N\backslash(N\backslash Si))\ \Rightarrow\ S}\,\downarrow L$$

((*saw j*) ((*help c*) ((*feed* (*ι hippos*)) *h*)))

Appendix B. Computer generated derivation of *accbab* in Mix

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{a\ \Rightarrow\ a}}{a, I\ \Rightarrow\ a}\,IL}{a, I, I\ \Rightarrow\ a}\,IL}{a, I, I, I\ \Rightarrow\ a}\,IL}{a, I, I, [\,]\ \Rightarrow\ a\uparrow I}\,\uparrow R \quad \dfrac{}{b\ \Rightarrow\ b}}{a, I, I, (a\uparrow I)\downarrow b\ \Rightarrow\ b}\,\downarrow L}{a, I, [\,], (a\uparrow I)\downarrow b\ \Rightarrow\ b\uparrow I}\,\uparrow R$$

$$\dfrac{\dfrac{\dfrac{}{S\ \Rightarrow\ S}}{S, I\ \Rightarrow\ S}\,IL}{S, [\,]\ \Rightarrow\ S\uparrow I}\,\uparrow R \quad \dfrac{\dfrac{\dfrac{}{a\ \Rightarrow\ a}}{a, I\ \Rightarrow\ a}\,IL}{a, [\,]\ \Rightarrow\ a\uparrow I}\,\uparrow R \quad \dfrac{}{b\ \Rightarrow\ b}}{a, (a\uparrow I)\downarrow b\ \Rightarrow\ b}\,\downarrow L}{S, (S\uparrow I)\downarrow a, (a\uparrow I)\downarrow b\ \Rightarrow\ b}\,\downarrow L$$

$$\dfrac{\dfrac{a, I, (b\uparrow I)\downarrow S, (a\uparrow I)\downarrow b, (S\uparrow I)\downarrow a, (a\uparrow I)\downarrow b\ \Rightarrow\ b}{a, [\,], (b\uparrow I)\downarrow S, (a\uparrow I)\downarrow b, (S\uparrow I)\downarrow a, (a\uparrow I)\downarrow b\ \Rightarrow\ b\uparrow I}\,\uparrow R \quad \dfrac{}{S\ \Rightarrow\ S}}{a, (b\uparrow I)\downarrow S, (b\uparrow I)\downarrow S, (a\uparrow I)\downarrow b, (S\uparrow I)\downarrow a, (a\uparrow I)\downarrow b\ \Rightarrow\ S}\,\downarrow L$$

References

Joachim Lambek. 1958. The mathematics of sentence structure. *American Mathematical Monthly*, 65:154–170. Reprinted in Buszkowski, Wojciech, Wojciech Marciszewski, and Johan van Benthem, editors, 1988, *Categorial Grammar*, Linguistic & Literary Studies in Eastern Europe volume 25, John Benjamins, Amsterdam, 153–172.

J. Lambek. 1961. On the Calculus of Syntactic Types. In Roman Jakobson, editor, *Structure of Language and its Mathematical Aspects, Proceedings of the Symposia in Applied Mathematics XII*, pages 166–178. American Mathematical Society, Providence, Rhode Island.

Michael Moortgat. 1995. Multimodal linguistic inference. *Journal of Logic, Language and Information*, 5:349–385. Also in *Bulletin of the IGPL*, 3(2,3):371–401, 1995.

Michael Moortgat. 1997. Categorial Type Logics. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, pages 93–177. Elsevier Science B.V. and The MIT Press, Amsterdam and Cambridge, Massachusetts.

Glyn Morrill and Oriol Valentín. 2010. Displacement Calculus. To appear in the Lambek Festschrift, special issue of *Linguistic Analysis*. http://arxiv.org/abs/1004.4181.

Glyn Morrill, Oriol Valentín, and Mario Fadda. 2009. Dutch Grammar and Processing: A Case Study in TLG. In Peter Bosch, David Gabelaia, and Jérôme Lang, editors, *Logic, Language, and Computation: 7th International Tbilisi Symposium, Revised Selected Papers*, number 5422 in Lecture Notes in Artificial Intelligence, pages 272–286, Berlin. Springer.

Glyn Morrill. 1990. Intensionality and Boundedness. *Linguistics and Philosophy*, 13(6):699–726.

Glyn Morrill. 1992. Categorial Formalisation of Relativisation: Pied Piping, Islands, and Extraction Sites. Technical Report LSI-92-23-R, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya.

Glyn V. Morrill. 1994. *Type Logical Grammar: Categorial Logic of Signs*. Kluwer Academic Press, Dordrecht.

Glyn V. Morrill. forthcoming. *Categorial Grammar: Logical Syntax, Semantics, and Processing*. Oxford University Press.

M. Pentus. 1992. Lambek grammars are context-free. Technical report, Dept. Math. Logic, Steklov Math. Institute, Moskow. Also published as ILLC Report, University of Amsterdam, 1993, and in *Proceedings Eighth Annual IEEE Symposium on Logic in Computer Science*, Montreal, 1993.

Stuart Shieber. 1985. Evidence Against the Context-Freeness of Natural Language. *Linguistics and Philosophy*, 8:333–343. Reprinted in Walter J. Savitch, Emmon Bach, William Marsh and Gila Safran-Naveh, editors, 1987, *The Formal Complexity of Natural Language*, D. Reidel, Dordrecht, 320–334.

J. van Benthem. 1991. *Language in Action: Categories, Lambdas, and Dynamic Logic*. Number 130 in Studies in Logic and the Foundations of Mathematics. North-Holland, Amsterdam. Revised student edition printed in 1995 by MIT Press.