The Hidden Structural Rules of the Discontinuous Lambek Calculus

Oriol Valentín
Universitat Politècnica de Catalunya

Abstract. The sequent calculus sL for the Lambek calculus L (1) has no structural rules. Interestingly, sL is equivalent to a multimodal calculus mL, which consists of the nonassociative Lambek calculus with the structural rule of associativity. This paper proves that the sequent calculus or hypersequent calculus hD of the discontinuous Lambek calculus 1 (7, 4 and 8), which like sL has no structural rules, is also equivalent to an ω-sorted multimodal calculus mD. More concretely, we present a faithful embedding translation (·)♯ between mD and hD in such a way that it can be said that hD absorbs the structural rules of mD.

1 The Discontinuous Lambek Calculus D and its Hypersequent Syntax

D is model-theoretically motivated, and the key to its conception is the class FreeDisp of displacement algebras. We need some definitions:

(1) Definition (Syntactical Algebra)

A syntactical algebra is a free algebra (L, +, 0, 1) of arity (2, 0, 0) such that (L, +, 0) is a monoid and 1 is a prime. I.e. L is a set, 0 ∈ L and + is a binary operation on L such that for all s1, s2, s3, s ∈ L,

s1 + (s2 + s3) = (s1 + s2) + s3 associativity
0 + s = s = s + 0 identity

The distinguished constant 1 is called a separator.

(2) Definition (Sorts)

The sorts of discontinuous Lambek calculus are the naturals 0, 1, . . . . The sort S(s) of an element s of a syntactical algebra (L, +, 0, 1) is defined by the morphism of monoids S to the additive monoid of naturals defined thus:

S(1) = 1
S(a) = 0 for a prime a ≠ 1
S(s1 + s2) = S(s1) + S(s2)

1 In [5] and [8], the term displacement calculus is used instead of Discontinuous Lambek Calculus as in [7] and [9].
I.e. the sort of a syntactical element is simply the number of separators it contains; we require the separator 1 to be a prime and the syntactical algebra to be free in order to ensure that this induction is well-defined.

(3) **Definition (Sort Domains)**

Where \((L, +, 0, 1)\) is a syntactical algebra, the sort domains \(L_i\) of sort \(i\) of generalized discontinuous Lambek calculus are defined as follows:

\[ L_i = \{ s | S(s) = i \}, i \geq 0 \]

(4) **Definition (Displacement Algebra)**

The displacement algebra defined by a syntactical algebra \((L, +, 0, 1)\) is the \(\omega\)-sorted algebra with the \(\omega\)-sorted signature \(\Sigma_D = (\oplus, \{ \otimes_i \} \in \omega, 0, 1)\) with sort functionality \(((i, j \rightarrow i + j), \mathbb{N}, 0, 1)\):

\[ (\{ L_i \}_{i \in \omega}, +, \{ \times_i \} \in \omega, 0, 1) \]

where:

| Operation | Description |
|-----------|-------------|
| \(+ : L_i \times L_j \rightarrow L_{i+j}\) | as in the syntactical algebra |
| \(\times_k : L_{i+1} \times L_j \rightarrow L_{i+j}\) | \(\times_k(s, t)\) is the result of replacing the \(k\)-th separator in \(s\) by \(t\) |

The sorted types of the discontinuous Lambek Calculus, \(D\), which we will define residuating with respect to the sorted operations in (4), are defined by mutual recursion in Figure 1.\(D\) types are to be interpreted as subsets of \(L\) and satisfy what we call the principle of well-sorted inhabitation:

- \(F_i ::= \mathcal{A}_i\) where \(\mathcal{A}_i\) is the set of atomic types of sort \(i\)
- \(F_0 ::= I\) Continuous unit
- \(F_1 ::= J\) Discontinuous unit
- \(F_{i\cdot j} ::= F_i \cdot F_j\) continuous product
- \(F_j ::= F_j \setminus F_{i\cdot j}\) continuous under
- \(F_i ::= F_{i\cdot j} / F_j\) continuous over
- \(F_{i\cdot j} ::= F_{i+1\cdot j} \ominus F_j\) discontinuous product
- \(F_j ::= F_j \downarrow F_{i\cdot j}\) discontinuous extract
- \(F_{i\cdot 1} ::= F_{i\cdot 1} \uparrow F_j\) discontinuous infix

**Fig. 1.** The sorted types of \(D\)
Principle of well-sorted inhabitation:

If \( A \) is a type of sort \( i \), \( \llbracket A \rrbracket \subseteq L_i \) Where \( \llbracket \cdot \rrbracket \) is the syntactical interpretation in a given displacement algebra w.r.t. a valuation \( v \). I.e. every syntactical inhabitant of \( \llbracket A \rrbracket \) has the same sort. The connectives and their syntactical interpretations are shown in Figures 1 and 2. This syntactical interpretation is called the standard syntactical interpretation. Given the functionalities of the operations with respect to which the connectives are defined, the grammar defining by mutual recursion the sets \( F_i \) of types of sort \( i \) on the basis of sets \( A_i \) of atomic types, and the homomorphic syntactical sort map \( S \) sending types to their sorts, are as shown in Figure 3. When \( A \) is an arbitrary type, we will frequently write in latin lower-case the type in order to refer to its sort \( S(A) \), i.e.:

\[
a \overset{\text{def}}{=} S(A)
\]

The syntactical sort map is to syntax what the semantic type map is to semantics: both homomorphisms mapping syntactic types to the datatypes of the respective components of their inhabiting signs in the dimensions of language in extension: form/signifier and meaning/signified.

\[
\begin{align*}
\llbracket I \rrbracket &= \{0\} & \text{continuous unit} \\
\llbracket J \rrbracket &= \{1\} & \text{discontinuous unit} \\
\llbracket A \rrbracket &\subseteq L_i \text{ for some } i \in \omega & A \in \mathcal{A}_i, \\
\llbracket A \cdot B \rrbracket &= \{s_1 + s_2 \mid s_1 \in \llbracket A \rrbracket &\& s_2 \in \llbracket B \rrbracket\} & \text{(continuous) product} \\
\llbracket A \setminus C \rrbracket &= \{s_2 \mid \forall s_1 \in \llbracket A \rrbracket, s_1 + s_2 \in \llbracket C \rrbracket\} & \text{under} \\
\llbracket C / B \rrbracket &= \{s_1 \mid \forall s_2 \in \llbracket B \rrbracket, s_1 + s_2 \in \llbracket C \rrbracket\} & \text{over} \\
\llbracket A \odot_k B \rrbracket &= \{\times_k(s_1, s_2) \mid s_1 \in \llbracket A \rrbracket &\& s_2 \in \llbracket B \rrbracket\} & k > 0 \text{ deterministic discontinuous product} \\
\llbracket A \upharpoonright_k C \rrbracket &= \{s_2 \mid \forall s_1 \in \llbracket A \rrbracket, \times_k(s_1, s_2) \in \llbracket C \rrbracket\} & k > 0 \text{ deterministic discontinuous infix} \\
\llbracket C \uparrow_k B \rrbracket &= \{s_1 \mid \forall s_2 \in \llbracket B \rrbracket, \times_k(s_1, s_2) \in \llbracket C \rrbracket\} & k > 0 \text{ deterministic discontinuous extract}
\end{align*}
\]

Fig. 2. Standard syntactical interpretation of D types

Observe also that (modulo sorting) \( (\setminus, \cdot, \div, \subseteq) \) and \( (\downarrow_k, \odot_k, \uparrow_k; \subseteq) \) are residuated triples:

\[
\begin{align*}
\llbracket B \rrbracket &\subseteq \llbracket A \setminus C \rrbracket \text{ iff } \llbracket A \cdot B \rrbracket \subseteq \llbracket C \rrbracket \text{ iff } \llbracket A \rrbracket \subseteq \llbracket C / B \rrbracket \\
\llbracket B \rrbracket &\subseteq \llbracket A \upharpoonright_k C \rrbracket \text{ iff } \llbracket A \odot_k B \rrbracket \subseteq \llbracket C \rrbracket \text{ iff } \llbracket A \rrbracket \subseteq \llbracket C \uparrow_k B \rrbracket
\end{align*}
\]

The types of D are sorted into types \( \mathcal{F}_i \) of sort \( i \) interpreted as sets of strings of sort \( i \) as shown in Figure 4 where \( k \in \omega^+ \).

If one wants to absorb the structural rules of a Gentzen sequent system in a substructural logic, one has to discover a convenient data structure for the antecedent and the
\( \mathcal{F}_i ::= \mathcal{A}_i \quad S(A) = i \quad \text{for } A \in \mathcal{A}_i \)

\( \mathcal{F}_0 ::= I \quad S(I) = 0 \)

\( \mathcal{F}_1 ::= J \quad S(J) = 1 \)

\( \mathcal{F}_{i \cdot j} ::= \mathcal{F}_i \cdot \mathcal{F}_j \quad S(A \cdot B) = S(A) + S(B) \)

\( \mathcal{F}_{i \setminus j} ::= \mathcal{F}_i \setminus \mathcal{F}_j \quad S(A \setminus C) = S(C) - S(A) \)

\( \mathcal{F}_{i / j} ::= \mathcal{F}_i / \mathcal{F}_j \quad S(C / B) = S(C) - S(B) \)

\( \mathcal{F}_{i \cdot j} ::= \mathcal{F}_{i \cdot j} \circ \mathcal{F}_j \quad S(A \circ B) = S(A) + S(B) - 1 \quad 1 \leq k \leq i + 1 \)

\( \mathcal{F}_{i / j} ::= \mathcal{F}_{i / j} \downarrow k \mathcal{F}_j \quad S(A \downarrow k C) = S(C) + 1 - S(A) \quad 1 \leq k \leq i + 1 \)

\( \mathcal{F}_{i \cdot j} ::= \mathcal{F}_{i \cdot j} \uparrow k \mathcal{F}_j \quad S(C \uparrow k B) = S(C) + 1 - S(B) \quad 1 \leq k \leq i + 1 \)

Fig. 3. Sorted D types, and syntactical sort map for D

\( \mathcal{F}_j \cdot \mathcal{F}_j \quad [A \setminus C] = \{ s_2 \mid \forall s_1 \in [A], s_1 + s_2 \in [C] \} \) under

\( \mathcal{F}_i \cdot \mathcal{F}_j \quad [C / B] = \{ s_1 \mid \forall s_2 \in [B], s_1 + s_2 \in [C] \} \) over

\( \mathcal{F}_{i \cdot j} \cdot \mathcal{F}_j \quad [A \cdot B] = \{ s_1 + s_2 \mid s_1 \in [A] \land s_2 \in [B] \} \) product

\( \mathcal{F}_0 ::= I \quad [I] = \{ 0 \} \) product unit

\( \mathcal{F}_j \cdot \mathcal{F}_j \quad [A \downarrow k C] = \{ s_2 \mid \forall s_1 \in [A], s_1 \times k s_2 \in [C] \} \) infix

\( \mathcal{F}_{i \cdot j} \cdot \mathcal{F}_j \quad [C \downarrow k B] = \{ s_1 \mid \forall s_2 \in [B], s_1 \times k s_2 \in [C] \} \) extract

\( \mathcal{F}_{i \cdot j} \cdot \mathcal{F}_j \quad [A \circ k B] = \{ s_1 \times k s_2 \mid s_1 \in [A] \land s_2 \in [B] \} \) disc. product

\( \mathcal{F}_1 ::= J \quad [J] = \{ 1 \} \) disc. prod. unit

Fig. 4. Types of the Discontinuous Lambek Calculus D and their interpretation
succe deport of sequents. We will now consider the Hypersequent syntax from [7]. The reason for using the prefix hyper in the term sequent is that the data-structure proposed is quite nonstandard. We define now what we call the set of types segments:

\[(7) \text{Definition (Type Segments)}\]

In hypersequent calculus we define the types segments \(SF_k\) of sort \(k\):

\[
SF_0 \ ::= \ A \quad \text{for} \quad A \in \mathcal{F}_0 \\
SF_a \ ::= \ \sqrt{A} \quad \text{for} \quad A \in \mathcal{F}_a \quad \text{and} \quad 0 \leq a = S(A)
\]

Types segments of sort 0 are types. But, types segments of sort greater than 0 are no longer types. Strings of types segments can form meaningful logical material like the set of hyperconfigurations, which we now define. The hyperconfigurations \(O\) are defined unambiguously by mutual recursion as follows, where \(\Lambda\) is the empty string and \([\ ]\) is the metalinguistic separator:

\[
O \ ::= \ A \\
O \ ::= \ A, O \quad \text{for} \quad S(A) = 0 \\
O \ ::= \ [], O \\
O \ ::= \ \sqrt{A}, O, \sqrt{A}, O, \sqrt{A}, O \\
\quad \text{for} \quad a = S(A) > 0
\]

The syntactical interpretation of \(\sqrt{A}, O, \sqrt{A}, O, \ldots, \sqrt{A}, O, \sqrt{A}\) consists of syntactical elements \(a_0 + \beta_1 + \alpha_1 + \cdots + \alpha_{n-1} + \beta_n + \alpha_n\) where \(a_0 + 1 + \alpha_1 + \cdots + \alpha_{n-1} + 1 + \alpha_n \in [A]\) and \(\beta_1 \in [A_1], \ldots, \beta_n \in [A_n]\). The syntax in which set \(O\) has been defined, is called string-based hypersequent syntax. An equivalent syntax for \(O\) is called tree-based hypersequent syntax which was defined in [4], [8].

In string-based notation the figure \(\overline{A}\) of a type \(A\) is defined as follows:

\[(8) \overline{A} = \begin{cases} A & \text{if} \ s(A) = 0 \\ \sqrt{A}, [], \sqrt{A}, [], \ldots, \sqrt{A}, [], \sqrt{A} & \text{if} \ s(A) > 0 \end{cases}\]

The sort of a hyperconfiguration is the number of metalinguistic separators it contains. Where \(\Gamma\) and \(\Phi\) are hyperconfigurations and the sort of \(\Gamma\) is at least 1, \(\Gamma_i \Phi\) \((k \in \omega^+)^k\) signifies the hyperconfiguration which is the result of replacing the \(k\)-th separator in \(\Gamma\) by \(\Phi\). Where \(\Gamma\) is a hyperconfiguration of sort \(i\) and \(\Phi_1, \ldots, \Phi_i\) are hyperconfigurations, the generalized wrap \(\Gamma \odot (\Phi_1, \ldots, \Phi_i)\) is the result of simultaneously replacing the successive separators in \(\Gamma\) by \(\Phi_1, \ldots, \Phi_i\) respectively. \(A(\Gamma)\) abbreviates \(A(\Gamma \odot (\lambda_1, \ldots, \lambda_i))\).

A hypersequent \(\Gamma \Rightarrow A\) comprises an antecedent hyperconfiguration in string-based notation of sorts \(i\) and a succedent type of sort \(i\). The hypersequent calculus for \(D\) is as shown in Figure 5 where \(k \in \omega^+\). Like \(L\), \(hD\) has no structural rules.

Morrill and Valentín (2010) [4] proves Cut-elimination for the \(k\)-ary discontinuous Lambe calculus, \(k > 0\). As a consequence \(D\), like \(L\), enjoys in addition the subformula property, decidability, and the finite reading property.

\[2\] Term which must not be confused with Avron’s hypersequents (1).
Fig. 5. Hypersequent calculus hD
2 hD: Absorbing the Structural Rules of a Sorted Multimodal Calculus

We consider now a sorted multimodal calculus mD with a set of structural rules EqD we present in the following lines. Figure 6 shows the logical rules of mD and Figure 7 shows the structural rules EqD integrated in mD. This sequent calculus is non standard in two senses. Types and structural terms are sorted. Moreover, there are two structural term constants which stand respectively for the continuous unit and discontinuous unit. Structural term constructors are of two kinds: ◦ (which stands for term concatenation) and ◦i (which stands for term wrapping at the i-th position, i ∈ ω*). Again, as in the case of sorted types, structural terms are defined by mutual recursion and the sort map is computed in a similar way (see (10)).

X[Y] denotes a structural term with a distinguished position occupied by the structural term Y. If A, X are respectively a type and a structural term, then a and x denote their sorts. We are interested in the cardinality of the set F of types of D and their structure. Consider the following lemma:

(9) Lemma
The set of types F is countably infinite iff the set of atomic types is countable. Moreover we have that:
\[ F = \bigcup_{i \in \omega} F_i \]
\[ F_i = (A_i)_{j \in \omega} \]

Proof. The proof can be carried out by coding in a finite alphabet the set of types F. Of course, it is crucial that the set of sorted atomic types forms a denumerable set. □

Let StructTermD[F] be the ω-sorted algebra over the signature ΣD = (◦) ∪ (◦i)_{i \in \omega}, I, J). The sort functionality of ΣD is:

\[ ((i, j \rightarrow i + j)_{i, j \in \omega}, (i + 1, j \rightarrow i + j)_{i, j \in \omega}, 0, 1) \]

Observe that the operations ◦ and ◦i’s (with i > 0) are sort polymorphic. In the following, we will abbreviate StructTermD[F] by StructTerm. The set of structural terms can be defined in BNF notation as follows:

(10) StructTerm0 ::= I
StructTerm1 ::= J
StructTerm_i ::= F_i
StructTerm_{i+j} ::= StructTerm_i ◦ StructTerm_j
StructTerm_{i+j} ::= StructTerm_{i+1} ◦ i StructTerm_j

It is clear that the sort of StructTerm, and the collections of set (A_i)_{j \in \omega} (i ∈ ω) are such that:

\[ S(StructTerm_i) = i \]
\[ S(A_i) = i \]
We realize that \textbf{StructTerm} looks like an \(\omega\)-sorted term algebra. This intuition is correct for the \(\omega\)-graduated set \(F\) with the collections \((A_{ij})_{j\in\omega}\) playing the role of an \(\omega\)-graduated set of a variables of an \(\omega\)-sorted term algebra \(T_{\Sigma^d}[X]\) with signature \(\Sigma^d\).

We need to define some important relations between structural terms.

(11) \textbf{Definition (Wrapping and Permutable Terms)}

Given the term \((T_1 \circ T_2) \circ T_3\), we say that:

(P1) \(T_2 \prec_{T_1} T_3\) iff \(i + t_2 - 1 < j\).

(P2) \(T_3 \prec_{T_2} T_2\) iff \(j < i\).

(O) \(T_2 \notin_{T_1} T_3\) iff \(i \leq j \leq i + t_2 - 1\).

Observe that in a term like \((T_1 \circ T_2) \circ T_3\), if (P1) or (P2) hold, (O) does not apply. Conversely, if (O) is applicable, neither (P1) nor (P2) hold. If \(T_2 \prec_{T_1} T_3\) (respectively \(T_3 \prec_{T_2} T_2\)), we say that \(T_2\) and \(T_3\) (respectively \(T_3\) and \(T_2\)) \textit{permute} in \(T_1\). Otherwise, if (O) holds, we say that \(T_2\) \textit{wraps} \(T_3\) in \(T_1\).

(12) \textbf{Example}

Suppose that \(T_1 = A\) where \(A\) is an arbitrary type of sort 3, and \(T_2, T_3\) are arbitrary structural terms. Let \(a_0 + 1 + a_1 + 1 + a_2 + 1 + a_3\) be an element of \([A]\) in a displacement model \(M\). Suppose \(S(T_2) = 3\). Consider now:

\[ (A \circ T_2) \circ T_3 \]

According to definition (11), \(T_2 \prec_A T_3\), for \(2 + S(T_2) - 1 = 4 < 5\). The intuition of this relation is the following. Interpreting in \(M\) we have that:

(13) \([A \circ T_2] \circ T_3\] = \(a_0 + 1 + a_1 + T_2 + a_2 + T_3 + a_3\)

We clearly see that the string \([T_2]\] precedes the occurrence of \([T_3]\]. Similarly, if we have \(T_3 \prec_A T_2\) in \((A \circ T_2) \circ T_3\), the occurrence of \([T_3]\] precedes \([T_2]\]. Finally, if \(T_2 \notin_A T_3\) then \([T_2]\] wraps \([T_3]\), i.e. \([T_3]\] is intercalated in \([T_2]\).

We define the following relation between structural terms \(\sim\):

(14) \(T \sim S\) iff \(S\) is the result of applying one structural rule to a subterm of \(T\)

\(\sim^+\) is defined to be the reflexive, symmetric and transitive closure of \(\sim\).

2.1 \textbf{The Faithful embedding translation \((\cdot)^\sharp\) between \textit{mD} and \textit{hD}}

We consider the following embedding translation from \(\textit{mD}\) to \(\textit{hD}\):

\[ (\cdot)^\sharp : \textit{mD} = (F, \text{StructTerm}, \rightarrow) \rightarrow \textit{hD} = (F, O, \Rightarrow) \]

\[ T \rightarrow A \quad \Rightarrow \quad (T)^\sharp \Rightarrow (A)^\sharp \]

\(\cdot\)\(\sharp\) is such that:

\[ A^\sharp = \overline{A} \text{ if } A \text{ is of sort strictly greater than 0} \]

\[ A^\sharp = A \text{ if } A \text{ is of sort 0} \]

\[ (T_1 \circ T_2)^\sharp = T_1^\sharp \circ T_2^\sharp \]

\[ (T_1 \circ T_2)^\sharp = T_1^\sharp | T_2^\sharp \]

\[ \lambda^\sharp = \lambda \]

\[ \lambda^\sharp = \emptyset \]
Fig. 6. The Logical rules of mD
Structural rules for units

- Continuous unit:

| Expression | Rule |
|------------|------|
| $T[X] \rightarrow A$ | $T[X] \rightarrow A$ |
| $T[\circ X] \rightarrow A$ | $T[X] \rightarrow A$ |
| $T[\circ I] \rightarrow A$ | $T[X] \rightarrow A$ |

- Discontinuous unit:

| Expression | Rule |
|------------|------|
| $T[X] \rightarrow A$ | $T[\circ X] \rightarrow A$ |
| $T[\circ I] \rightarrow A$ | $T[X] \rightarrow A$ |

Continuous associativity

| Expression | Rule |
|------------|------|
| $X[(T_1 \circ T_2) \circ T_3] \rightarrow D$ | $ASSC_c$ |
| $X[T_1 \circ (T_2 \circ T_3)] \rightarrow D$ | $ASSC_c$ |

Split-wrap

| Expression | Rule |
|------------|------|
| $T_1[T_2 \circ T_3] \rightarrow D$ | $SW$ |
| $T_1[(\circ T_2) \circ T_3] \rightarrow D$ | $SW$ |
| $T_1[T_2 \circ T_3] \rightarrow D$ | $SW$ |
| $T_1[(\circ T_2) \circ T_3] \rightarrow D$ | $SW$ |

Discontinuous associativity $T_2 \bowtie_{T_1} T_3$

| Expression | Rule |
|------------|------|
| $S[T_1 \circ (T_2 \circ J_{T_1})] \rightarrow C$ | $Assc_1$ |
| $S[(T_1 \circ T_2) \circ J_{T_1}] \rightarrow C$ | $Assc_2$ |

Mixed permutation 1 case $T_2 \prec_{T_1} T_3$

| Expression | Rule |
|------------|------|
| $S[(T_1 \circ T_2) \circ J_{T_1}] \rightarrow C$ | $MixPerm_1$ |
| $S[(T_1 \circ T_2) \circ J_{T_1}] \rightarrow C$ | $MixPerm_1$ |

Mixed permutation 2 case $T_3 \prec_{T_1} T_2$

| Expression | Rule |
|------------|------|
| $S[(T_1 \circ T_2) \circ J_{T_1}] \rightarrow C$ | $MixPerm_2$ |
| $S[(T_1 \circ T_2) \circ J_{T_1}] \rightarrow C$ | $MixPerm_2$ |

Fig. 7. Structural Rules of mD
Collapsing the structural rules

Let us see how the structural rules are absorbed in \( hD \). We show here that structural postulates of \( mD \) collapse into the same textual form when they are mapped through \((\cdot)^\sharp\). Later we will see that:

If \( T \sim S \) then \( T^\sharp = S^\sharp \)

Moreover will see that for every \( A, B, C \in F \) the following hypersequents are provable in \( hD \):

\[
(15) \quad \text{– Continuous associativity}
\]

\[
\frac{A \bullet (B \bullet C)}{\Rightarrow (A \bullet B) \bullet C} \quad \text{and} \quad \frac{(A \bullet B) \bullet C}{\Rightarrow A \bullet (B \bullet C)}
\]

\[
(16) \quad \text{– Mixed associativity}
\]

\[
\frac{A \circ_i (B \circ_j C)}{\Rightarrow (A \circ_i B) \circ_{i+j-1} C} \quad \text{and} \quad \frac{(A \circ_i B) \circ_{i+j-1} C}{\Rightarrow A \circ_i (B \circ_j C)}
\]

\[
(15) \quad \text{– Mixed permutation}
\]

\[
\frac{A \circ_i B \circ_j C}{\Rightarrow (A \circ_{j-b+1} C) \circ_i C} \quad \text{and} \quad \frac{(A \circ_{j-b+1} C) \circ_i C}{\Rightarrow (A \circ_i B) \circ_j C}
\]

If we have that \( B \triangleleft A \):

\[
\frac{(A \circ_i B) \circ_j C}{\Rightarrow (A \circ_j C) \circ_{i+c-1} C (A \circ_j C) \circ_{i+c-1} C \Rightarrow (A \circ_i B) \circ_j C)
\]

\[
(16) \quad \text{– Split wrap}
\]

\[
\frac{A \bullet B}{\Rightarrow (A \bullet J) \circ_{i+1} B} \quad \text{and} \quad \frac{(A \bullet J) \circ_{i+1} B}{\Rightarrow A \bullet B}
\]

\[
(16) \quad \text{– Continuous unit and discontinuous unit}
\]

\[
\frac{A \circ_i J}{\Rightarrow A} \quad \text{and} \quad \frac{A}{\Rightarrow A \circ_i J} \quad \text{and} \quad \frac{I \circ A}{\Rightarrow A} \quad \text{and} \quad \frac{J \circ A}{\Rightarrow I \circ A}
\]

That \( hD \) absorbs the rules is proved in the following theorem:

(16) **Theorem (hD Absorption of EqD Structural Rules)**

For any \( T, S \in \text{StructTerm} \), if \( T \sim S \) then \((T)^\sharp = (S)^\sharp \).

**Proof.** We define a useful notation for vectorial types which will help us to prove the theorem. Where \( A \) is an arbitrary type of sort greater than 0:
Note that $\mathcal{A} = \mathcal{A}_0$. Now, consider arbitrary types $A$, $B$ and $C$. As usual we denote their sorts respectively by $a$, $b$ and $c$. We have then:

- Continuous associativity:
  \[
  \begin{align*}
  &((A \circ B) \circ C)^{\mathcal{A}} = (\mathcal{A}, \mathcal{B}, \mathcal{C}) = \mathcal{A}, \mathcal{B}, \mathcal{C} \\
  &(A \circ (B \circ C))^{\mathcal{A}} = (\mathcal{A}, (\mathcal{B}, \mathcal{C}) = \mathcal{A}, \mathcal{B}, \mathcal{C}
  \end{align*}
  \]

- Discontinuous associativity: Suppose that $B \triangleleft_{\mathcal{A}} C$

We have that:

\[
\begin{align*}
\overline{B}_i | \mathcal{C} &= B_0^{i-1}, \mathcal{C}, B_j^{b} \\
\overline{A}_i | (\overline{B}_i | \mathcal{C}) &= \overline{A}_0^{i-1}, B_0^{i-1}, \mathcal{C}, B_j^{b}, \mathcal{A}_i
\end{align*}
\]

On the other hand, we have that:

\[
\overline{A}_i | \overline{B} = \overline{A}_0^{i-1}, \overline{B}, \mathcal{A}_i = \overline{A}_0^{i-1}, B_0^{i-1}, \underbrace{\mathcal{A}_j}_{(i+j-1)-th}, B_j, \mathcal{A}_i
\]

It follows that:

\[
(A \circ (B | C))^{\mathcal{A}} = \overline{A}_0^{i-1}, B_0^{i-1}, \mathcal{C}, B_j^{b}, \mathcal{A}_i
\]

Summarizing:

\[
\begin{align*}
&((A \circ (B \circ C))^{\mathcal{A}} = \overline{A}_0^{i-1}, B_0^{i-1}, \mathcal{C}, B_j^{b}, \mathcal{A}_i \\
&((A \circ (B \circ C))^{\mathcal{A}} = \overline{A}_0^{i-1}, B_0^{i-1}, \mathcal{C}, B_j^{b}, \mathcal{A}_i
\end{align*}
\]

Hence:

\[
(A \circ (B \circ C))^{\mathcal{A}} = ((A \circ (B \circ C))^{\mathcal{A}}
\]

For the case $(A \circ (B \circ C))^{\mathcal{A}}$, if one puts $k = i + j - 1$ one gets $j = k - i + 1$. Therefore, changing indices: we have that:

\[
((A \circ (B \circ C))^{\mathcal{A}} = (A \circ (B \circ (C \circ j+1)))^{\mathcal{A}}
\]

This ends the case of discontinuous associativity.

- Mixed permutation:

There are two cases: $B \triangleleft_{\mathcal{A}} C$ or $C \triangleleft_{\mathcal{A}} B$. We consider only the first case, i.e. $B \triangleleft_{\mathcal{A}} C$. The other case is analogous. Let us see $((A \circ (B \circ C))^{\mathcal{A}}$:

\[
\overline{A}_i | \overline{B} = \overline{A}_0^{i-1}, \overline{B}, \mathcal{A}_j^{k-1}, \underbrace{\mathcal{A}_k}_{j-th}
\]
We have therefore:

\[ j = k - 1 + b \text{ iff } k = j - b + 1 \]

\[ ((A \circ_j B) \circ_j C)^\# = \overline{A_i^{d^{-1}}}, \overline{B_i^{d^{-1}}}, \overline{C_i^{d^{-1}}}, \overline{A_k^{d^{a^1}}} \]

Hence:

\[ (A \circ_{j - b + 1} C)^\# = \overline{A_i^{d^{-1}}}, \overline{B_i^{d^{-1}}}, \overline{C_i^{d^{-1}}}, \overline{A_k^{d^{a^1}}} \]

It follows that:

\[ ((A \circ_j B) \circ_j C)^\# = \overline{A_i^{d^{-1}}}, \overline{B_i^{d^{-1}}}, \overline{C_i^{d^{-1}}}, \overline{A_k^{d^{a^1}}} \]

Summarizing:

\[
\begin{cases}
((A \circ_j B) \circ_j C)^\# = \overline{A_i^{d^{-1}}}, \overline{B_i^{d^{-1}}}, \overline{C_i^{d^{-1}}}, \overline{A_k^{d^{a^1}}} \\
((A \circ_{j - b + 1} C) \circ_j B)^\# = \overline{A_i^{d^{-1}}}, \overline{B_i^{d^{-1}}}, \overline{C_i^{d^{-1}}}, \overline{A_k^{d^{a^1}}} 
\end{cases}
\]

Hence

\[ ((A \circ_j B) \circ_j C)^\# = ((A \circ_{j - b + 1} C) \circ_j B)^\# \]

Putting \( i = j - b + 1 \) we have that \( j = i + b - 1 \). Hence:

\[ ((A \circ_j C) \circ_j B)^\# = ((A \circ_{i + b - 1} C) \circ_j B)^\# \]

This ends the case of mixed permutation.

- Split-wrap:

  We have:

  \[ ((A \circ_i B) \circ_{i + 1} B)^\# = (\overline{\overline{A}}, \overline{\overline{B}}) = \overline{\overline{A}}, \overline{\overline{B}} \]

  \[ ((J \circ_i B) \circ_{i + 1} A)^\# = (\overline{\overline{A}}, \overline{\overline{B}}) = \overline{\overline{A}}, \overline{\overline{B}} \]

  Hence:

  \[ ((A \circ_i J) \circ_{i + 1} B)^\# = (A \circ B)^\# \]

  \[ (J \circ_i A)^\# = (\overline{\overline{A}}, \overline{\overline{B}}) = \overline{\overline{A}}, \overline{\overline{B}} \]  

  This ends the case of split-wrap.

- Units:

\[
\begin{align*}
(A \circ I)^\# &= \overline{A} = (A \circ I)^\# \\
(J \circ_1 A)^\# &= (\overline{\overline{A}}, \overline{\overline{A}}) = \overline{\overline{A}}, \overline{\overline{A}} = (A \circ J)^\#
\end{align*}
\]

We recall that types play the role of variables of structural terms. Now, we have seen that structural rules for arbitrary type variables collapse into the same textual form. This result generalizes to arbitrary structural terms by simply using type substitution.
More concretely, we have proved that: if $T \sim S$ (i.e. $S$ is the result of applying a single structural rule to $T$) then $T^\sharp = S^\sharp$. Suppose we have $T \sim S$ (we omit the trivial case $T \sim T$). We have then a chain:

$$T := T_1 \sim T_2 \sim \cdots \sim T_{i-1} \sim T_i =: S$$ for $i \geq 2$

Applying $(\cdot)^\sharp$ to each $T_k \sim T_{k+1}$ $(1 \leq k \leq i - 1)$ we have proved that:

$$(T_k)^\sharp = (T_{k+1})^\sharp$$

We have therefore a chain of identities:

$$(T)^\sharp = (T_1)^\sharp = (T_2)^\sharp = \ldots = (T_i)^\sharp = (S)^\sharp$$

This completes the proof.

□

We will now prove the associativity theorems of $h\textbf{D}$ displayed in (15). Other theorems corresponding to the structural postulates of $m\textbf{D}$ have similar proofs.

- Continuous associativity is obvious as in the Lambek calculus. The only difference is that types are sorted and in our notation the antecedent of hypersequents have the vectorial notation.
- Discontinuous associativity: we suppose that $B \not\Rightarrow A \circ_i C$. The following hypersequents are provable:

$$\vdash (A \circ_i B) \circ_{i+j-1} C \Rightarrow A \circ_i (B \circ_j C)$$

And:

$$\vdash A \circ_i (B \circ_j C) \Rightarrow (A \circ_i B) \circ_{i+j-1} C$$

By the previous lemma the identity $\vdash (B|C) = (A|B)_{i+j-1}$ holds. We have the two following hypersequent derivations:

$$\vdash A \Rightarrow (A|B)_{i+j-1} \circ_j R$$

and

$$\vdash B \Rightarrow B \circ_j C \Rightarrow C \circ_j L$$

and
Let $R$ and $S$ be arbitrary structural terms. The following holds:

$$R \sim^* S \text{ iff } (R)^\sharp = (S)^\sharp$$

**Proof.** We have already seen in (16) the only if case, which is the fact that $\text{hD}$ absorbs the $\text{Eq}_\text{D}$ structural rules. The if case is more difficult and needs some technical machinery from sorted universal algebra. For details, see [9]. □

(19) **Lemma** ($\cdot)^\sharp$ is an Epimorphism

For every $A \in O$ there exists a structural term $T_A$ such that:

$$(T_A)^\sharp = A$$

**Proof.** This can be proved by induction on the structure of hyperconfigurations. We define recursively $T_A$ such that $(T_A)^\sharp = A$:

- Case $A = \lambda$ (the empty tree): $T_A = 1$.
- Case where $A = \lambda, \Gamma$: $T_A = A \circ T_\Gamma$, where by induction hypothesis (i.h.) $(T_\Gamma)^\sharp = \Gamma$.
- Case where $A = [], \Gamma$: $T_A = \lambda \circ T_\Gamma$, where by i.h. $(T_\Gamma)^\sharp = \Gamma$.
- Case $A = \overline{A} \circ (A_1, \cdots, A_n), A_{a+1}$. By i.h. we have:

$$(T_A)^\sharp = A_i \text{ for } 1 \leq i \leq a + 1$$

$$T_A = (A \circ_1 T_A) \circ T_\Delta \text{ if } a = 1$$

$$T_A = ((\cdots ((A \circ_1 T_A) \circ_1 a T_A) \cdots) \circ_1 a + d_a T_A_{a + 1} \cdots T_{a + 1} \circ T_{a + 1} \cdots T_{a + 1} T_{a + 1} \circ T_{a + 1} \cdots T_{a + 1}) \circ T_{a + 1} \cdots T_{a + 1} \circ T_{a + 1} \cdots T_{a + 1} \text{ if } a > 1$$

□

By induction on the structure of $\text{StructTerm}$, we have the following intuitive result on the relationship between structural contexts and hypercontexts:

(20) \(T[S])^\sharp = T^\sharp (S)^\sharp\)

These two technical results we have seen above are necessary for the proof of the faithful embedding translation ($\cdot)^\sharp$ of theorem (24). We prove now an important theorem which is crucial for the mentioned theorem (24).

---

3 In fact there exists an infinite set of such structural terms.
(21) **Theorem (Visibility for Extraction in StructTerm)**

Let $T[A]$ be a structural term with a distinguished occurrence of type $A$. Suppose that:

$$(T[A])^\sharp = \Delta \mid \iota$$

where $\Delta \in O$ and $A \in \mathcal{F}$. Then $A$ is visible for extraction in $T[A]$, i.e. there exist a structural term $T'$ and an index $i$ such that:

$$T[A] \sim^* T' \circ_i A$$

**Proof.** Let $T_\Delta$ be a structural term such that $(T_\Delta)^\sharp = \Delta$. This is possible by lemma (19). We have $(T_\Delta \circ_i A)^\sharp = \Delta \mid \iota$. We then $(T_\Delta \circ_i A)^\sharp = (T[A])^\sharp$. By the equivalence theorem (18) it follows that $T[A] \sim^* T_\Delta \circ_i A$. Put $T' := T_\Delta$. We are done. □

(22) **Theorem (Uniqueness of Extractability)**

Suppose that $T[A] \sim S \circ_i A$ and $T[A] \sim S' \circ_j A$, where $A$. Then:

$$S \sim^* S'$$

$$i = j$$

**Proof.** We have that $(S \circ_i A)^\sharp = \Delta \mid \iota = (S' \circ_j A)^\sharp$. Hence $i = j$ and $(S)^\sharp = (S')^\sharp$. By theorem (18), $S \sim^* S'$. We are done. □

Theorems (21) and (22) will be crucial for the proof of the $(\cdot^\sharp)$ embedding theorem (24).

Before proving theorem (24), it is worth seeing what is the intuition behind the structural rules of $\mathcal{E}_D$. This intuition is exemplified by a constructive proof of theorem (21).

**Proof.** Constructive proof of theorem (21): By induction on the structural complexity of $T[A]$: The cases are as follows:

i) $T[A] = A$.

We put $T' = \bar{J}$ and hence :

$$T[A] \sim^* \bar{J} \circ_1 A$$

ii) $T[A] = S[A] \circ R$.

By induction hypothesis (i.h.), $S[A] \sim^* S' \circ_k A$ for some $k > 0$. We have the following equational derivation:

$$T[A] \sim^* (S' \circ_k A) \circ R$$

$$\sim^* (\bar{J} \circ R) \circ_1 (S' \circ_k A) \quad \text{by SW}$$

$$\sim^* ((\bar{J} \circ R) \circ_1 S') \circ_2 A \quad \text{by Ascc}_d$$

$$\sim^* (S' \circ R) \circ_2 A \quad \text{by SW}$$
iii) $T[A] = S \circ R[A]$

By i.h. $R[A] \sim^* R'[\ell] A$ for some term $R'$ and $k > 0$. It follows that:

$$
T[A] \sim^* S \circ (R'[\ell] A) \\
\sim^* (S \circ)_{S[S]} (R'[\ell] A) \quad \text{by SW} \\
\sim^* ( (S \circ)_{\pi S[S]i} R')_{S[S]} \quad \text{by Assc} \\
\sim^* (S \circ R')_{S[S]} \quad \text{by SW}
$$

iv) $T[A] = S[A] \circ_i R$ for some term $S[A]$, $R$ and $i > 0$.

By i.h. $S[A] \sim^* S'[\ell] A$ for some $S'$ and $i > 0$. We derive the following equation:

$$
T[A] \sim^* (S'[\ell] A) \circ_i R
$$

If $R = J$ we are done. Suppose that $R \not= J$. In this case $A$ must permute with $R$ in $S'$, i.e. $A \lessdot S' R$ or $R \lessdot S' A$, for otherwise $(T[A])^\sharp = \Delta | A$ would not hold. Without loss of generality, let us suppose that $A \lessdot S' R$. In that case we have:

$$
T[A] \sim^* (S' \circ_{\pi S[S]} (R') \circ_i A) \quad \text{by MixPerm1}
$$

Hence $A$ is permutated to right periphery in $T[A]$.

v) $T[A] = S \circ R[A]$ for some terms $S$ and $R[A]$ and $i > 0$. By i.h. $R[A] \sim^* R'[\ell] A$. Then:

$$
T[A] \sim^* S \circ (R'[\ell] A) \\
\sim^* (S \circ R')_{S[S]} \quad \text{by Assc}
$$

□

(23) **Remark**

Interestingly, the constructive proof for extractability does not use continuous associativity. Therefore, a priori a non-associative discontinuous Lambek calculus could be considered. This remark needs further study.

(24) **Theorem (Faithfulness of $\cdot^\sharp$ Embedding Translation)**

Let $A$, $X$ and $\Delta$ be respectively a type, a structural term and a hyperconfiguration. The following statements hold:

i) If $\vdash_{mD} X \rightarrow A$ then $\vdash_{hD} (X)^\sharp \rightarrow A$

ii) For any $X$ such that $(X)^\sharp = \Delta$, if $\vdash_{hD} \Delta \Rightarrow A$ then $\vdash_{mD} X \rightarrow A$

**Proof.**

i) Logical rules in $mD$ translate without any problem to $hD$. We need recall only that if $X$ and $Y$ are structural terms then $(X \circ Y)^\sharp = (X)^\sharp, (Y)^\sharp$ and $(X \circ_i Y)^\sharp = (X)^\sharp | (Y)^\sharp$.

Structural rules in $mD$ collapse in the same textual form as theorem (16) proves.

Finally, the Cut rule has no surprise. This proves i).

ii) This part of the theorem becomes easy if we use the following four facts:
• Lemma (19) which states that for any hyperconfiguration \( \Delta \) there is a structural term \( T_\Delta \) such that \( (T_\Delta)_{\downarrow} = \Delta \).

• The fact (20) we stated before which gives the relationship between structural terms and hypercontexts \( (T[A])_{\downarrow} = T^\Delta(A) \).

• Theorem (18).

• Theorem (21).

The proof is by induction on the length of \( hD \) derivations. The three first facts prove the induction of all the rules but the right rule of the connectives \( \uparrow_i \). Suppose the last rule of a \( hD \) derivation is \( \uparrow_i R \):

\[
\Delta|_i \overline{A} \Rightarrow B \\
\Delta \Rightarrow B^i_\uparrow A
\]

Let \( T[A] \) be such that \( (T[A])_{\downarrow} = \Delta|_i \overline{A} \). We know by induction hypothesis that \( \vdash_{mD} T[A] \Rightarrow B \). By the last fact of above, i.e. theorem (21) of visibility of extraction, since \( (T[A])_{\downarrow} = \Delta|_i \overline{A} \), we know there exist \( T' \) and \( i \) such that \( T[A] \sim T' \circ_i A \). It follows that in \( mD \):

\[
\begin{align*}
T[A] & \rightarrow B \\
\text{: Sequence of structural rules} \\
T' \circ_i A & \rightarrow B \\
T' & \rightarrow B^i_\uparrow A
\end{align*}
\]

Hence, \( \vdash_{mD} T' \rightarrow B^i_\uparrow A \). And \( T' \) is in fact \( T_\Delta \), and therefore \( (T')_{\downarrow} = \Delta \). Moreover, for any \( S \) such that \( (S)_{\downarrow} \sim T' \), we have that applying a finite number of structural rules we obtain the \( mD \) provable sequent \( S \rightarrow B^i_\uparrow A \), and of course \( (S)_{\downarrow} = \Delta \). This completes the proof of ii).

\( \square \)

(25) Example

Let \( B, D, E, C, A \) five arbitrary atomic types of sort respectively 2, 2, 1, 0 and 0. The following two derivations have the following end-sequent and end-hypersequent:

\[
\begin{align*}
\vdash_{mD} ((B \circ_2 D \circ_2 E) \circ_3 (J \circ C \setminus A)) \rightarrow ((B \circ_1 D) \circ_3 E) \uparrow_3 C \\
\vdash_{hD} \sqrt{B^2_\uparrow A, \overline{D}, \sqrt{B^2_\uparrow A, [], C \setminus A, \sqrt{B^2_\uparrow A, \overline{E}, \sqrt{B^2_\uparrow A \Rightarrow ((B \circ_1 D) \circ_3 E) \uparrow_3 C}}}}
\end{align*}
\]

The above multimodal sequents are in correspondence through the mapping \( (\cdot)_{\downarrow} \).

Derivations (26) and (27) illustrate theorem (24). Notice the sequence of structural rules in derivation (26) in order to extract type \( C \).
It is not a priori a trivial task to find out a set of structural rules. The equivalent multimodal calculus equivalent to a multimodal calculus with the structural rules of $\mathcal{E}$. The faithful embedding translation (\(\delta\)) between $\mathbf{mD}$ and $\mathbf{hD}$ is then, we think, a remarkable discovery. The equivalent multimodal calculus $\mathbf{mD}$ gives $\mathbf{D}$ the Moot’s powerful proof net machinery almost for free (see \([3]\)). It must be noticed that this proof net theory approach for $\mathbf{D}$ is completely different from the one in \([6]\). Finally, the discovery of $\mathbf{mD}$ can be very useful to investigate new soundness and completeness results for $\mathbf{D}$ (see \([9]\)).

3 Conclusions

It is not a priori a trivial task to find out a set of structural rules $\mathcal{E}$ that makes the hypersequent calculus $\mathbf{hD}$ equivalent to a multimodal calculus with the structural rules of $\mathcal{E}$. The faithful embedding translation (\(\delta\)) between $\mathbf{mD}$ and $\mathbf{hD}$ is then, we think, a remarkable discovery. The equivalent multimodal calculus $\mathbf{mD}$ gives $\mathbf{D}$ the Moot’s powerful proof net machinery almost for free (see \([3]\)). It must be noticed that this proof net theory approach for $\mathbf{D}$ is completely different from the one in \([6]\). Finally, the discovery of $\mathbf{mD}$ can be very useful to investigate new soundness and completeness results for $\mathbf{D}$ (see \([9]\)).

References

1. A. Avron. Hypersequents, Logical Consequence and Intermediate Logic form Concurrency. Annals of Mathematics and Artificial Intelligence, 4:225–248, 1991.
2. Joachim Lambek. The mathematics of sentence structure. American Mathematical Monthly, 65:154–170, 1958. Reprinted in Buszkowski, W., W. Marciszewski, and J. van Benthem, editors, 1988, Categorial Grammar, Linguistic & Literary Studies in Eastern Europe volume 25, John Benjamins, Amsterdam, 153–172.
3. Richard Moot. Proof nets for display logic. CoRR, abs/0711.2444, 2007.
4. G. Morrill and O. Valentín. Displacement Calculus. Linguistic Analysis, 36(1-4):167–192, 2010.
5. G. Morrill, O. Valentín, and M. Fadda. Dutch grammar and processing: A case study in tlg. In Gabelaia Bosch and Lang, editors, *Logic, Language, and Computation: 7th International Tbilisi Symposium, Revised Selected Papers*, number 5422 in Springer Lecture Notes in Artificial Intelligence, pages 272–286, Berlin, 2009. Springer.

6. Glyn Morrill and Mario Fadda. Proof Nets for Basic Discontinuous Lambek Calculus. *Logic and Computation*, pages 239–256, 2008.

7. Glyn Morrill, Mario Fadda, and Oriol Valentín. Nondeterministic Discontinuous Lambek Calculus. In Jeroen Geertzen, Elias Thijssse, Harry Bunt, and Amanda Schiffrin, editors, *Proceedings of the Seventh International Workshop on Computational Semantics, IWCS-7*, pages 129–141. Tilburg University, 2007.

8. Glyn Morrill, Oriol Valentín, and Mario Fadda. The Displacement Calculus. *Journal of Logic, Language and Information*, 20(1):1–48, 2011. Doi 10.1007/s10849-010-9129-2.

9. Oriol Valentín. *Theory of Discontinuous Lambek Calculus*. PhD thesis, Universitat Autònoma de Barcelona, Barcelona, 2012.