Proof nets for the Displacement calculus

Richard Moot

Proof nets for the Displacement calculus

Richard Moot

CNRS (LaBRI)

1 Introduction

The Displacement calculus was introduced by Morrill, Valentín & Fadda (2011) as an extension of the Lambek calculus with discontinuous operators. These discontinuous connectives allow the Displacement calculus to solve a large number of problems with the Lambek calculus. Examples of the phenomena treated by Morrill et al. (2011) include discontinuous idioms (such as "ring up" and "give the cold shoulder"), quantifier scope, extraction (including pied-piping) and gapping.

This paper extends earlier work by Morrill & Fadda (2008), Moot (2014) and Valentín (2014), combining the strengths of these different approaches while at the same time diminishing the inconveniences. Notably, it is the first proof net calculus which does not operate by translation into some other logic, but provides proof nets for the Displacement calculus directly.

2 The Displacement calculus

The presentation of the Displacement calculus closely follows the natural deduction calculus used by Morrill et al. (2011). String terms are built over a countably infinite alphabet of variables (for readability, we will often used natural language words as if they were variables), a special separator symbol "$\mathbf{1}$", where string concatenation is denoted by "+" (a binary, associative infix operator on string terms). As usual, ϵ denotes the empty string. The *sort* of a string term is the number of occurrences of the separator "$\mathbf{1}$".

I use lower-case roman letters p, q ... for atomic string terms (for enhanced readability, I will often use the standard convention of using words from the lexicon in the place of such atomic string terms), lower-case roman letters a, b, ... for string terms without separator symbols and lower-case greek letters α, β, ... for strings containing any number of separator symbols. So the string term $p + \mathbf{1} + q + \mathbf{1} + r$ is a string of sort 2 with three atomic subterms.

The key to the Displacement calculus is the wrap operator $\alpha \times_k \beta$. There is some minor variation in the definition of this operator: sometimes (Morrill et al. 2011) k is either the constant ">" or the constant "<" (in which case α is of sort greater than zero and the denotation of the term replaces respectively the first and the last occurrences of $\mathbf{1}$ in α by β. Sometimes (Morrill 2011) k is an integer (between 1 and the sort of α) and $\alpha \times_k \beta$ replaces the kth separator

in α by β. The equations below given the definition of "$\times_k$".

$$(1) \qquad (a + \mathbf{1} + \alpha) \times_> \beta =_{def} a + \beta + \alpha$$

$$(2) \qquad (\alpha + \mathbf{1} + a) \times_< \beta =_{def} \alpha + \beta + a$$

$$(3) \qquad (a_1 + \mathbf{1} + \ldots + a_n + \mathbf{1} + \alpha) \times_n \beta =_{def} a_1 + \mathbf{1} + \ldots + a_n + \beta + \alpha$$

Where the Lambek calculus connectives get their meaning with respect to concatenation "+", the discontinuous connectives of the Displacement calculus get their meaning with respect to "$\times_k$" (this entails different connectives for different values of k). The standard interpretation of the Lambek calculus connectives for string models, with "+" denoting concatenation, is the following.

$$(4) \qquad |A \setminus C| =_{def} \{\beta \,|\, \forall\alpha \in |A|, \alpha + \beta \in |C|\}$$

$$(5) \qquad |C \,/\, B| =_{def} \{\alpha \,|\, \forall\beta \in |B|, \alpha + \beta \in |C|\}$$

$$(6) \qquad |A \bullet B| =_{def} \{\alpha + \beta \,|\, \alpha \in |A| \wedge \beta \in |B|\}$$

The discontinuous connectives of the Displacement calculus use "$\times_k$" instead of "+" (we present only the connectives for $>$ here).

$$(7) \qquad |A \downarrow_> C| =_{def} \{\beta \,|\, \forall\alpha \in |A|, \alpha \times_> \beta \in |C|\}$$

$$(8) \qquad |C \uparrow_> B| =_{def} \{\alpha \,|\, \forall\beta \in |B|, \alpha \times_> \beta \in |C|\}$$

$$(9) \qquad |A \odot_> B| =_{def} \{\alpha \times_> \beta \,|\, \alpha \in |A| \wedge \beta \in |B|\}$$

We can further unfold these definitions, using Definition 1 for "$\times_>$" to obtain.

$$(10) \qquad |A \downarrow_> C| =_{def} \{\beta \,|\, \forall(a + \mathbf{1} + \alpha) \in |A|, a + \beta + \alpha \in |C|\}$$

$$(11) \qquad |C \uparrow_> B| =_{def} \{(a + \mathbf{1} + \alpha) \,|\, \forall\beta \in |B|, a + \beta + \alpha \in |C|\}$$

$$(12) \qquad |A \odot_> B| =_{def} \{a + \beta + \alpha \,|\, (a + \mathbf{1} + \alpha) \in |A| \wedge \beta \in |B|\}$$

Given these definitions, the meaning of $A \downarrow_> C$ is defined as the set of expressions which select a circumfix A, which wraps around the string denoted by $A \downarrow_> C$ to form an expression C. Similarly, $C \uparrow_> B$ extracts a B formula not occurring after a separator.

2.1 Formulas and sorts

We have already defined the sort of a string term as the number of occurrences of the separator constant "**1**". The *sort* of a formula corresponds to the number of separators "**1**" occurring in its denotation. That is, a formula of sort n is assigned a string term of the form $a_0 + \mathbf{1} + \ldots + \mathbf{1} + a_n$ (with all a_i of sort 0 according to our notational convention). For a given grammar, its signature defines the sort of all atomic formulas occurring in the grammar. We assume throughout that the atomic formulas s, n, np, pp have sort 0 (some other atomic formulas, such as *inf* when used for Dutch verb clusters, have sort 1).

Table 1 shows how to compute the sort of complex formulas. All subformulas of a formula are assigned a sort, so when we compute $s(C \,/\, B)$ using its entry

$s(A \bullet B) = s(A) + s(B)$	$s(A \odot B) = s(A) + s(B) - 1$	$_{s(A)\geq 1}$
$s(A \setminus C) = s(C) - s(A)$	$s(A \downarrow C) = s(C) + 1 - s(A)$	$_{s(A)\geq 1}$
$s(C \,/\, B) = s(C) - s(B)$	$s(C \uparrow B) = s(C) + 1 - s(B)$	$_{s(C)\geq s(B)}$

Table 1. Computing the sort of a complex formula given the sort of its immediate subformulas

in Table 1 we know that $s(C) \geq s(B)$, because if not, then $s(C \,/\, B)$ would be less than zero and therefore not a valid (sub)formula (similar constraints can be derived from the other implications, eg. we can show that $s(C \uparrow B) \geq 1$).

As an example, following Morrill et al. (2011), we can assign a discontinuous lexical entry like "give the cold shoulder" the lexical formula $(np \setminus s) \uparrow_{>} np$ and string term $gave + \mathbf{1} + the + cold + shoulder$ (of the required sort 1).

2.2 Natural deduction rules

Figures 1 and 2 give the natural deduction rules for the Lambek calculus and for the left wrap rules respectively (the other wrap rules follow the same pattern). The left wrap rules of Figure 2 correspond rather closely to the interpretation of the formulas given in Definitions 10 to 12.

$$\frac{\alpha : A \quad \gamma : A \setminus C}{\alpha{+}\gamma : C}\ \setminus E \qquad \frac{\begin{array}{c}[\alpha : A]^i \\ \vdots \\ \alpha{+}\gamma : C\end{array}}{\gamma : A \setminus C}\ \setminus I_i$$

$$\frac{\gamma : C \,/\, B \quad \beta : B}{\gamma{+}\beta : C}\ /E \qquad \frac{\begin{array}{c}[\beta : B]^i \\ \vdots \\ \gamma{+}\beta : C\end{array}}{\gamma : C \,/\, B}\ /I_i$$

$$\frac{\delta : A \bullet B \qquad \begin{array}{c}[\alpha : A]^i \quad [\beta : B]^i \\ \vdots \\ \gamma[\alpha{+}\beta] : C\end{array}}{\gamma[\delta] : C}\ \bullet E_i \qquad \frac{\alpha : A \quad \beta : B}{\alpha{+}\beta : A \bullet B}\ \bullet I$$

Fig. 1. Proof rules – Lambek calculus

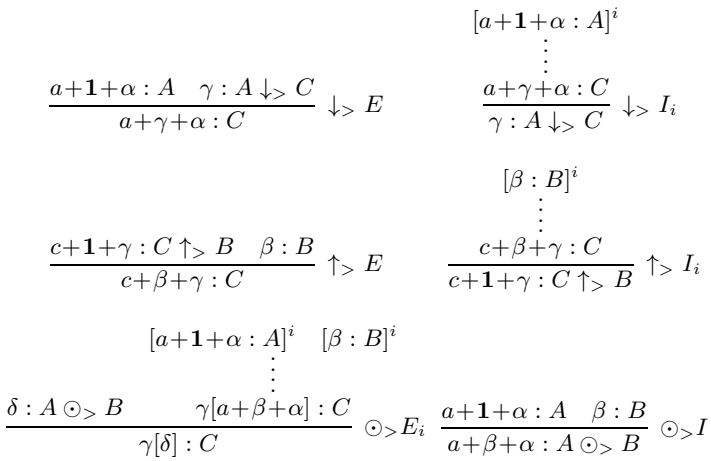


Fig. 2. Proof rules — leftmost infixation,extraction

3 Proof nets

One of the goals of proof search in type-logical grammars is to enumerate all possible readings for a given sentence. The bureaucratic aspects of the sequent calculus proof search make it hard to use sequent calculus directly for this goal, since sequent calculus allows a great number of inessential rule permutations. The situation for natural deduction is somewhat better, since the proof rules correspond directly to steps in meaning composition, even though there is still a large number of possible rule permutations for the $\bullet E$ and $\odot_k E$ rules.

Proof nets are a way of representing proofs which removes the "bureaucratic" aspects of sequent proofs and simplifies the product rules of Lambek calculus natural deduction. One of the open questions of Morrill (2011) is whether the Displacement calculus has a proof net calculus.

Valentín (2014) provides a translation of the Displacement calculus to a multimodal system. However, this system uses a rather large set of structural rules and these rules are defined modulo equivalence classes, which makes their use in existing multimodal theorem provers (Moot 2007) difficult. In this section, I will extend the proof net calculus for the Lambek calculus of Moot & Puite (2002) to the Displacement calculus. I will, in particular, provide an efficiently checkable correctness condition in the form of graph contractions.

3.1 Links

Figures 3 and 4 show the links for Displacement calculus proof structures. Each link connects three formulas to a central node. The formulas written above the central node of a link are the *premisses* of the link, the formulas written below it are its *conclusions*. The linear order of both the premisses and the conclusions of a link is important.

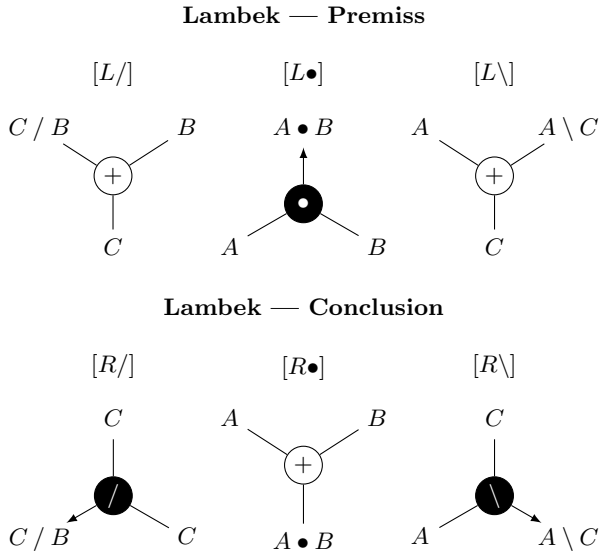


Fig. 3. Links for the Lambek calculus connectives of the Displacement calculus

We distinguish between par links, where the central node is filled black, and tensor links, where the central node is not filled (this is the familiar tensor/par distinction of multiplicative linear logic). Par nodes are further distinguished by an arrow pointing to the main formula of the link.

3.2 Proof structures

A *proof structure* is a set of formulas and a set of links such that.

1. each link instantiates one of the links shown in Figures 3 and 4 (for specific values of A, B, C and k),
2. each formula is the premiss of at most one link,
3. each formula is the conclusion of at most one link.

Formulas which are not the premiss of any link are the conclusions of the proof structure. Formulas which are not the conclusion of any link are the hypotheses of the proof structure (the word "conclusion" is overloaded: we talk about conclusions of proofs, conclusions of rules, conclusions of links and conclusions of proof structures; when the intended use is clear from the context, I will often simply use the word "conclusion" without further qualification). The *inputs* of a proof structure are its hypotheses and the active conclusions of its par links (that is, the conclusions of all par links in the proof structure except, for the implications, the one with the arrow); we will call the inputs which are not hypotheses the *auxiliary inputs* of a proof structure.

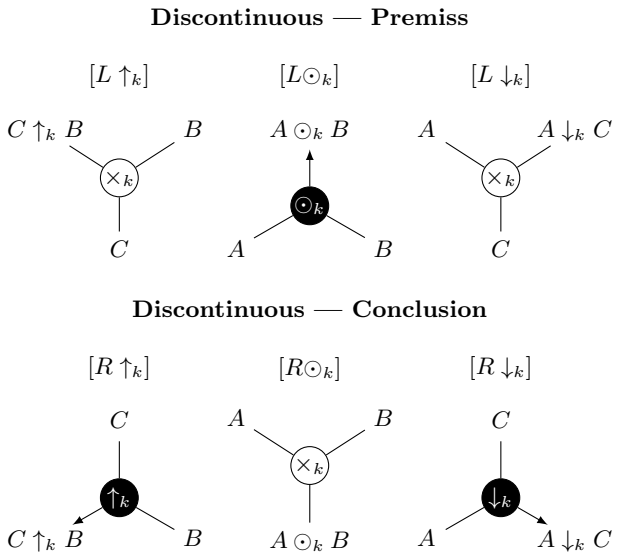


Fig. 4. Links for the discontinuous connectives of the Displacement calculus

To construct a proof structure for a given sequent $A_1, \ldots, A_n \vdash C$, we unfold the A_i as premisses and C as a conclusion. This will provide a proof structure with (atomic) conclusions other than C and (atomic) premisses other than the A_i. We identify these atomic hypotheses with atomic conclusions (of the same atomic formula) until we obtain a proof structure of $A_1, \ldots, A_n \vdash C$. This can fail if an atomic formula has more occurrences as a hypothesis than as a conclusion (as it should, since such sequents are underivable).

Figure 5 gives an unfolding for the sentence "Mary rang everyone up", a sentence with the discontinuous idiom "rang up" and a non-peripheral quantifier "everyone", following lexical entries of Morrill et al. (2011). Figure 6 shows (on the left of the figure) one of the possibilities for connecting the atomic formulas.

Not all proof structures correspond to natural deduction proofs. Proof structures which correspond to natural deduction proofs are *proof nets*. Of course, defining proof nets this way is not very satisfactory: we want to have a condition which, given a proof structure tells us whether or not this proof structure is a proof net using only properties of the proof structure itself.

3.3 Abstract Proof Structures

The general strategy we follow to define a correctness criterion for proof structures is as follows: we first simplify by removing some of the information which is irrelevant for deciding correctness to obtain abstract proof structures, then specify the correctness condition on these abstract proof structures, using a graph

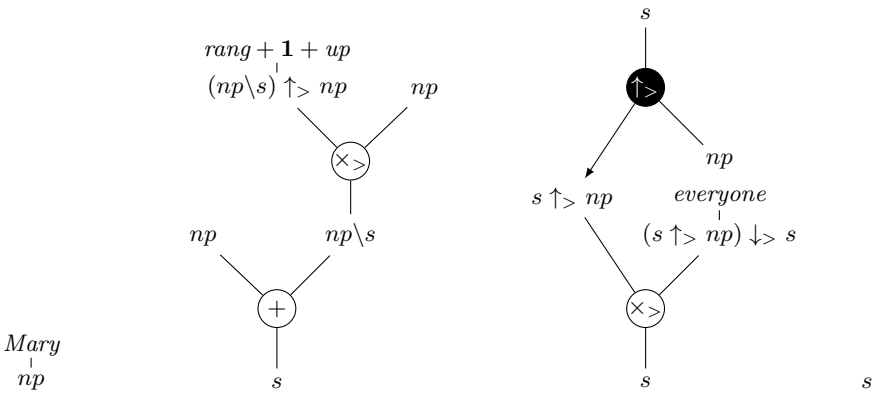


Fig. 5. Unfolding for the sentence "Mary rang everyone up".

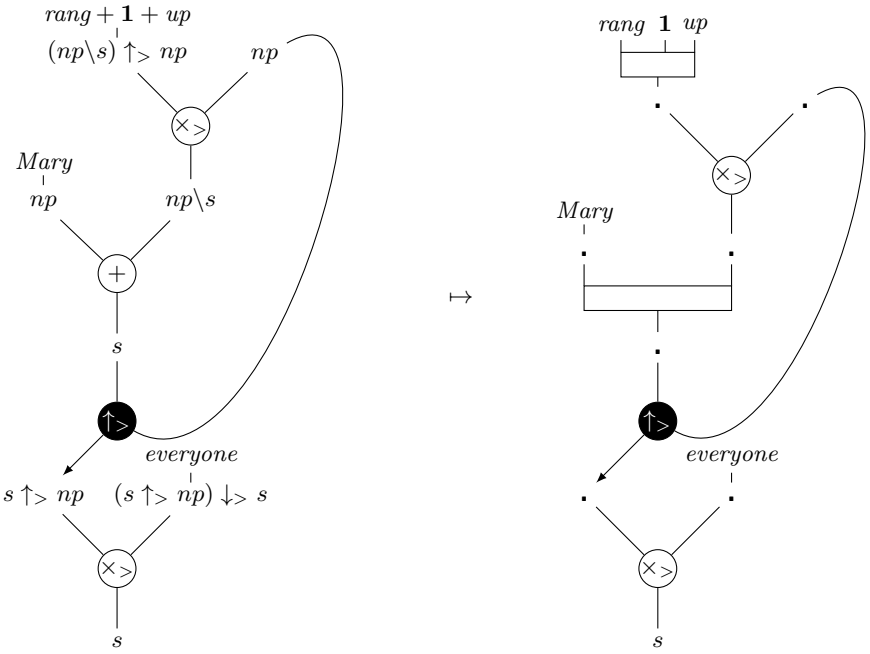


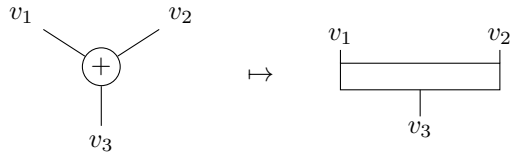
Fig. 6. Proof structure (left) and abstract proof structure (right) for the unfolding of "Mary rang everyone up" shown in Figure 5.

contraction criterion, generalizing the proof nets of the Lambek calculus from Moot & Puite (2002).

Tensor trees and combs A *tensor tree* is a connected, acyclic set of tensor links (to be more precise, the underlying undirected graph must by acyclic and connected). A single vertex is a tensor tree. Given an (abstract) proof structure, its tensor trees are the maximal substructures which are tensor trees; this is simply the forest we obtain when we remove all par links from a proof structure. The proof structure of Figure 6 has two tensor trees.

A *comb* is a link with any number of premisses and a single conclusion. None of the premisses of the comb can be identical to its conclusion. The general conditions on links prevent premisses from being connected more than once as a premiss of a comb. The premisses of combs, as links in general, are linearly ordered. Premisses of a comb can be hypotheses of the proof structure, the conclusions of a link or the special constant $\mathbf{1}$. The *sort* of a comb, that is the sort assigned to its conclusion, is the sum of the sorts of its premisses (the constant $\mathbf{1}$ is of sort 1). Combs play the same role as tensor trees do for Moot & Puite (2002): they allow us to go back a forth between sequents $\Gamma \vdash C$ and combs with premisses Γ and conclusion C. Given a comb, we will refer to subsequences of its premisses as prefixes, postfixes, etc., and assign them sorts as well.

Translating a proof structure to an abstract proof structure To translate a proof structure $\mathcal{P}$ to an abstract proof structure $\mathcal{A}$, we define a function, $\mathcal{P} \mapsto \mathcal{A}$, which replaces "+" links by 2-premiss combs as follows



which leaves all other links the same and which replaces the vertices/formulas of $\mathcal{P}$ as shown in Figure 7. The only slight complication is for the input formulas (lexical our auxiliary). Proof structures are defined as ways of connecting formulas, but for formulating correctness we need to know about the strings denoted by these formulas, for example, about their position relative to other formulas, separator symbols or the left/rightmost position. Another way of seeing this is that we need to replace sorted variables α (such as those assigned to hypotheses) by variables $p_0 + \mathbf{1} + \ldots + \mathbf{1} + p_n$, with each p_i of sort 0 (such a strategy is already implicitly used for the natural deduction rules for $/I$, $\backslash I$, $\bullet E$, $\uparrow_k I$, $\downarrow_k I$ and $\odot_k E$, that is the natural deduction rules corresponding to the par links). As shown in Figure 7, auxiliary inputs separate the path leaving the par link by adding n new subpaths (this appears somewhat odd, but is required for the correct behaviour of the contractions when sorts are greater than 0, as we will see below). Because of the sorts of the formulas, the par links for $\downarrow_k$ and $\odot_k$ necessarily involve at least one such split, though the other par links need not.

3.4 Contractions

Structural contractions Figure 8 shows the structural contractions. The "+" contraction composes two combs, combining the premisses by a simple left-to-right

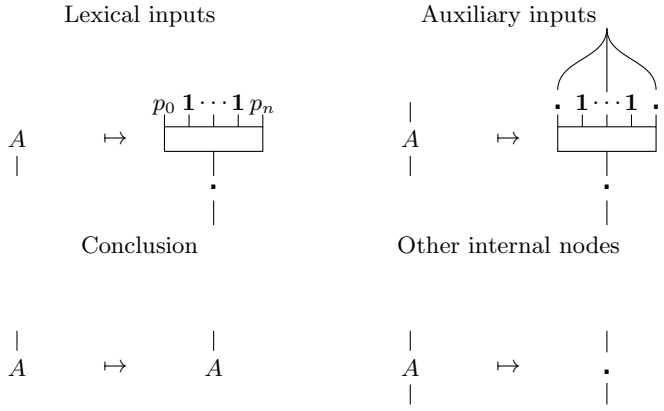


Fig. 7. Conversion to abstract proof structures for vertices/formulas.

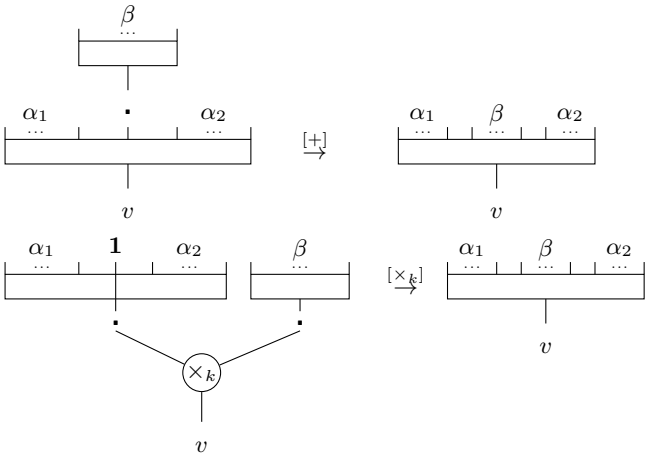


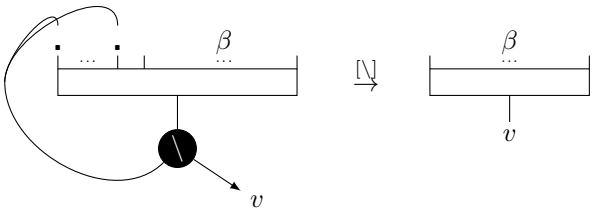
Fig. 8. Structural contractions

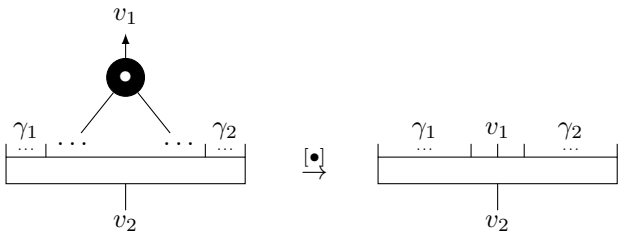
traversal. It is worth mentioning some immediate corollaries of this contraction here: first, we simply eliminate trivial combs (containing a single premiss and a single conclusion, that is, when β contains only a single premiss) when their conclusion is the premiss of another comb, and, second, the structural contractions contract tensor trees to unique combs (this is no longer guaranteed once we add the synthetic connectives, as discussed in Section 4).

The wrap operation "$\times_k$" reflects the wrap operation on strings on the level of abstract proof structures, it inserts β at the separator indicated by k: if k is ">", the α_1 must be of sort 0 (we replace the first separator by β) and if k is "<" α_2 must be of sort 0 (we replace the last separator by β).

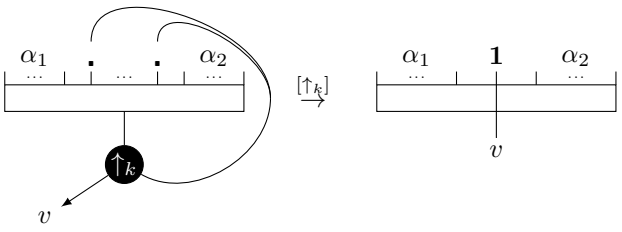
Note that α_1, α_2 and β are allowed to have zero premisses.

Logical contractions The logical contractions ensure the logical symmetry of the connectives in the calculus. Each par rule has its own contraction. The contraction for $\backslash$, shown below, essentially checks whether the string term of the premiss is equivalent to $p_0 + \mathbf{1} + \ldots + \mathbf{1} + p_n + \beta$ and withdraws the hypothesis $p_0 + \mathbf{1} + \ldots + \mathbf{1} + p_n$ (where n is the sort of the withdrawn formula in the corresponding $\backslash I$ rule) to reduce to β (the / contraction is left-right symmetric).

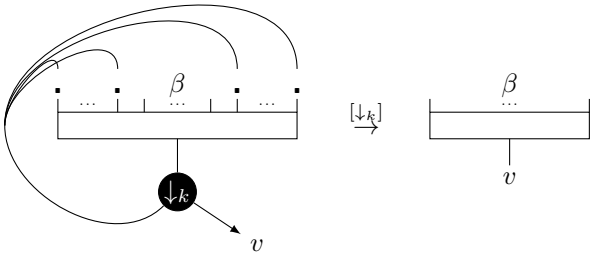




The contraction for $\uparrow_k$ essentially checks that its auxiliary input is an infix of the appropriate sort.

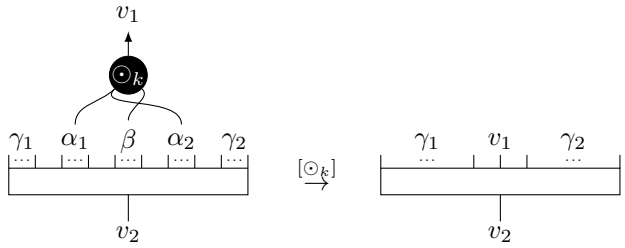


Depending on k, there are restrictions of the sorts: for ">", α_1 must be of sort 0 (that is, all premisses of the comb to the left β are of sort 0), for "<", α_2 must be of sort 0 (that is, all premisses of the comb to the right β are of sort 0), for $k = n$, α_1 is a prefix of sort $n - 1$ (that is, the sorts of the premisses of the comb to the left β sum to $n - 1$).



If k is ">", the premisses to the left of β are of sort 0. If k is "<", the premisses to the right of β are of sort 0. If $k = n$, the premisses to the left of β are of sort $n - 1$. This contraction looks odd until we realize that we are dealing with a circumfix operation and that, as a consequence the subformula A of a formula $A \downarrow_k B$ denotes a discontinuous circumfix with corresponding string $\alpha_1 + \mathbf{1} + \alpha_2$ (look back to the introduction rule for $\downarrow_k$ on the top right of Figure 2 for comparison).

The contraction for $\odot_k$ generalizes the contraction for $\bullet$. Whereas the contraction for $A \bullet B$ verifies the strings of the subformulas A and B are adjacent, the contraction for $A \odot_k B$ verifies whether the string $\alpha_1 + \mathbf{1} + \alpha_2$ of A is circumfixed around the string β of B. The sorts of α_1, α_2 and β depend on k and on the sorts of A and B, exactly as for the other rules (in the rule below, the labels α_1, α_2 and β represent sequences of vertices which are premisses of the comb and conclusions of the $\odot_k$ rule).



As an example of how we can use the contraction criterion to verify whether a proof structure is a proof net, the abstract proof structure of Figure 6 can be contracted first by using a "+" and a "$\times_>$" contraction to produce the abstract proof structure shown on the left of Figure 9, then by performing the "$\uparrow_>$" and "$\times_>$" contractions as indicated, giving a proof of "Mary rang everyone up".

Brief remarks on complexity It is easy to see the given contraction calculus is confluent and that each of the contraction steps reduces the total number of links in the structure. Therefore, even a naive implementation of this contraction calculus checks whether or not a given abstract proof structure with n links contracts to a comb in $O(n^3)$ steps, simply by repeatedly traversing the links in the graph to find contractible configurations and contracting them once they are found (we can likely improve upon this worst case, but I will leave this to further research).

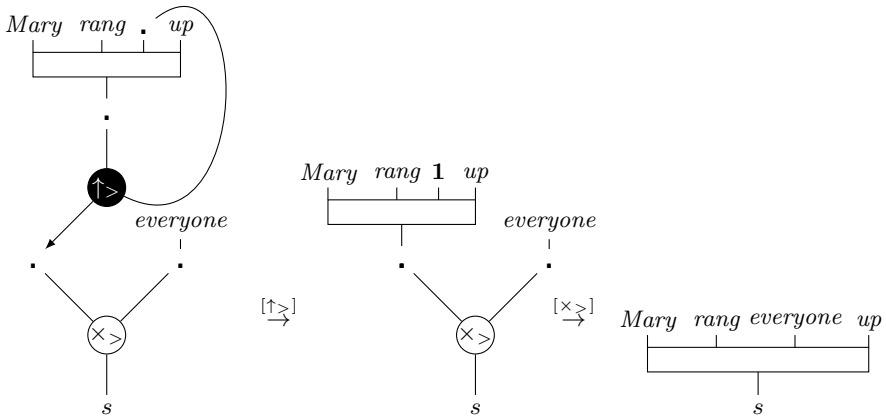


Fig. 9. Contractions for the abstract proof structure for the sentence "Mary rang everyone up" shown on the right of Figure 6.

In particular, this shows NP-completeness of the Displacement calculus. NP-hardness follows from NP-completeness of the Lambek calculus (Pentus 2006) and we can show it is in NP since we can verify in polynomial time whether or not a candidate proof (that is, a proof structure) in the Displacement calculus is a proof (that is, a proof net).

3.5 Correctness of the calculus

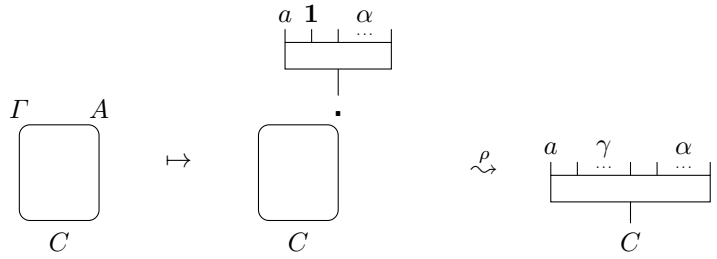
We show that the two definitions of proof net, contractibility and corresponding to a natural deduction proof coincide, thereby establishing that the contraction criterion is correct.

Lemma 1. *Let δ be a Displacement calculus natural deduction proof of $\alpha_1 : A_1, \ldots, a_n : A_n \vdash \gamma : C$. There is a proof net with the same hypotheses whose abstract proof structure contracts to a $\gamma : C$.*

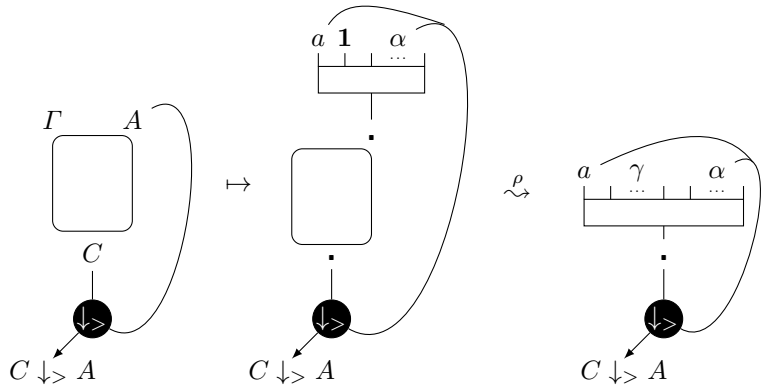
Proof This is a simple induction on the length of the proof. Axioms $\alpha : A \vdash \alpha : A$ correspond directly to proof nets with the required combs. Otherwise, we proceed by case analysis on the last rule of the proof. Each logical rule correspond to adding a link to the proof net(s) given by induction hypothesis and a contraction to the sequence of contractions for the abstract proof structure. We show only the case for $\downarrow_>$. In this case, the last rule looks as follows.

$$\dfrac{\begin{array}{c}[a{+}\mathbf{1}{+}\alpha : A]^i \\ \vdots\ \delta \\ a{+}\gamma{+}\alpha : C\end{array}}{\gamma : A \downarrow_> C}\ \downarrow_> I_i$$

Induction hypothesis gives use a proof net of $\Gamma, a{+}\mathbf{1}{+}\alpha : A \vdash a{+}\gamma{+}\alpha : C$. That is we are in the situation shown below, with the proof structure shown below on the right, the corresponding abstract proof structure in the middle, for which we are given a sequence of reductions ρ to a comb $a{+}\gamma{+}\alpha : C$. We have simply spelled out the definition of proof net of $\Gamma, a{+}\mathbf{1}{+}\alpha : A \vdash a{+}\gamma{+}\alpha : C$.



Adding the par link for $\downarrow_>$ to the above proof net produces to following proof structure, which contracts using the same sequence of contractions ρ as follows.

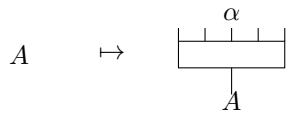


Simply performing the contraction for $\downarrow_>$ to the final abstract proof structure produces a comb of $\gamma : C \downarrow_> A$ and hence a proof net of $\Gamma \vdash \gamma : C \downarrow_> A$ as required. □

Lemma 2. *Let Π be a proof net of $\alpha_1 : A_1, \ldots, a_n : A_n \vdash \gamma : C$, that is a proof net with hypotheses $\alpha_1 : A_1, \ldots, a_n : A_n$ and conclusion C and an abstract proof structure contracting to γ using contractions ρ. There is a natural deduction proof of $\alpha_1 : A_1, \ldots, a_n : A_n \vdash \gamma : C$.*

Proof We proceed by induction on the number of logical contractions l in the sequence ρ (this number is equal to the number of par links in the structure).

If there are no logical contractions ($l = 0$), then there are only structural contractions and our proof net contains only tensor links. We proceed by induction on the number t of tensor links. If there are no tensor links ($t = 0$), we have an axiom and its abstract proof structure is a comb by definition.

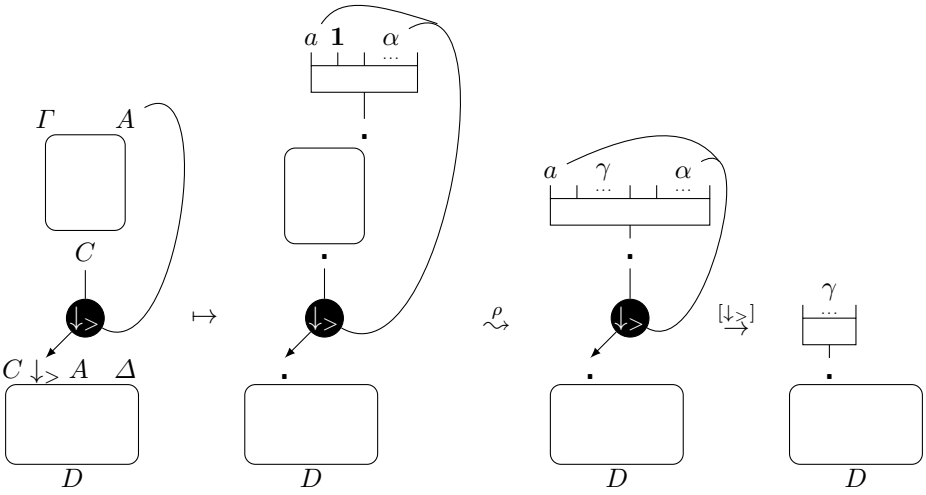


This directly gives us the natural deduction proof $\alpha : A \vdash \alpha : A$.

If there are tensor links ($t > 0$), then either one of the hypotheses or the conclusion of the proof structure must be the main formula of its link (this is easy to see since if none of the leaves is the main formula of its link, then the proof structure contains only introduction rules for $\bullet$ and $\odot_k$ and therefore the conclusion is the main formula of its link). Suppose a proof net has a leaf which is the main formula of its link and suppose this formula is $A \downarrow_> C$ (the cases of other formulas being main formulas, and of a conclusion of the proof net being the main formula are similar). Then, since all tensor trees contract to combs, we can apply the induction hypothesis to the two structures obtained by removing the tensor link and obtain proofs π_1 of $\Gamma \vdash a + \mathbf{1} + \alpha : A$ and π_2 of $\Delta, a + \gamma + \alpha : C \vdash \delta : D$ (technically, we have a proof with hypothesis $\gamma' : C$ and use substitution of the proof with conclusion $a + \gamma + \alpha : C$ shown below). We can combine these proofs as follows.

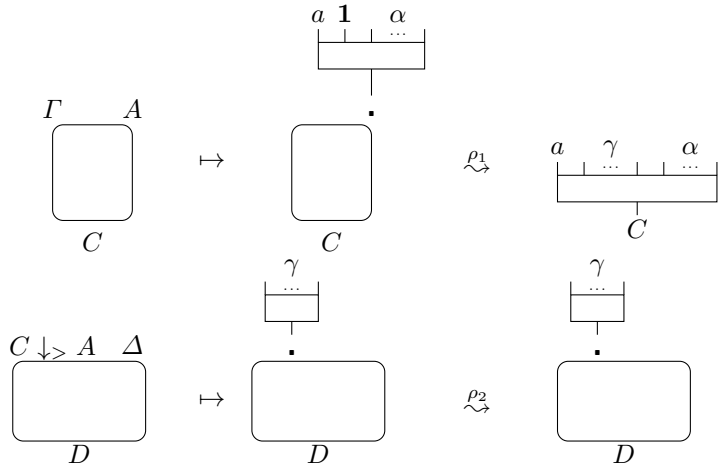
$$\dfrac{\begin{array}{c}\Gamma \\ \vdots\ \pi 1 \\ a + \mathbf{1} + \alpha : A \quad \gamma : A \downarrow_> C\end{array}}{\begin{array}{c}a + \gamma + \alpha : C \\ \vdots\ \pi 2 \\ \delta : D\end{array}} \downarrow_> E \quad \Delta$$

If the sequence ρ has logical contractions ($l > 0$), we look at the last such contraction and proceed by case analysis. If the last contraction is the $\downarrow_>$ contraction, our proof net and contraction sequence look as follows.



The initial proof structure is shown above of the left and its corresponding abstract proof structure to its immediate right (note that vertex A has been replaced by $a{+}\mathbf{1}{+}\alpha$, since it is an auxiliary input, corresponding to a withdrawn hypothesis in the natural deduction proof). The reduction sequence is of the form ρ, followed by the $\downarrow_>$ contraction, possibly followed by a number of structural contractions (not displayed in the figure above).

When we remove the par link from the figure above, we are in the following situation. All contractions from ρ are either fully in the abstract proof structure shown below at the top of the picture or fully in the abstract proof structure shown below at the bottom of the picture, so ρ splits naturally in ρ_1 and ρ_2.



We need to show that $\Gamma, \Delta \vdash \delta : D$ (where $\delta : D$ is the comb). Since we have two proof nets with strictly shorter sequences of contractions, we can apply the induction hypothesis for proofs π_1 of $a{+}\mathbf{1}{+}\alpha : A, \Gamma \vdash a{+}\gamma{+}\alpha : C$ and π_2 of $\gamma : C \downarrow_> A, \Delta \vdash \delta : D$. We can combine these two proofs into a proof of $\Gamma, \Delta \vdash \delta : D$ as follows.

$$\dfrac{\dfrac{[a{+}\mathbf{1}{+}\alpha : A]_i \quad \Gamma}{\vdots\ \pi_1}}{\dfrac{a{+}\gamma{+}\alpha : C}{\gamma : C \downarrow_> A}\downarrow_> I_i \quad \Delta}$$
$$\vdots\ \pi_2$$
$$\delta : D$$

The other cases are similar and easily verified. □

Theorem 1. *A proof structure is a* proof net *iff its abstract proof structure contracts to a comb.*

Proof Immediate from Lemma 1 and Lemma 2. □

4 Extension to Other Connectives

One of the benefits of the current calculus is that it extends easily to other connectives, such as the unary/bracket connectives of Morrill (2011, Chapter 5) (although incorporating the treatment of parasitic gapping of Section 5.5 would require a considerable complication of the proof theory).

The synthetic connectives of Morrill et al. (2011) require us to extend our methodology somewhat: as currently formulated the proof net calculus produces a single comb for each proof net. When adding the synthetic connectives, we can introduce a separation marker in a way which is only partially specified by the premiss of the rule. For example, the denotation of (leftmost) split $\check{}A$, shown below, is the set of strings obtained by inserting a separator symbol at any place before other separator symbols (if any), and therefore the introduction rule for this connective doesn't produce a unique string term.

$$|\check{}A| =_{def} \{a + \mathbf{1} + \alpha \mid a + \alpha \in |A|\} \tag{13}$$

$$|\hat{}A| =_{def} \{a + \alpha \mid a + \mathbf{1} + \alpha \in |A|\} \tag{14}$$

This moves us to a system where a tensor tree contracts to a set of combs (or, alternatively, a partially specified comb). Apart from this, it is not hard to add links and contractions for the synthetic connectives. For example, the contraction for $\check{}$ can be obtained from the contraction for $\uparrow_k$ by removing the links to the auxiliary hypothesis: instead of replacing the auxiliary hypothesis by $\mathbf{1}$ (which defines the position of the insertion point uniquely), there will be multiple, non-confluent ways to matching the contraction and to insert the separator symbol. For lack of space, we will not develop these ideas further here.

5 Conclusion

We have presented a proof net calculus for the Displacement calculus and shown its correctness. This is the first proof net calculus which models the Displacement calculus directly and not by some sort of translation into another formalism. The proof net calculus opens up new possibilities for parsing and proof search with the Displacement calculus.

References

Casadio, C., Coecke, B., Moortgat, M. & Scott, P., eds (2014), *Categories and Types in Logic, Language, and Physics: Essays dedicated to Jim Lambek on the Occasion of this 90th Birthday*, number 8222 *in* 'Lecture Notes in Artificial Intelligence', Springer.

Moot, R. (2007), Filtering axiom links for proof nets, *in* L. Kallmeyer, P. Monachesi, G. Penn & G. Satta, eds, 'Proccedings of Formal Grammar 2007'.

Moot, R. (2014), Extended Lambek calculi and first-order linear logic, *in* Casadio, Coecke, Moortgat & Scott (2014), pp. 297–330.

Moot, R. & Puite, Q. (2002), 'Proof nets for the multimodal Lambek calculus', *Studia Logica* **71**(3), 415–442.

Morrill, G. (2011), *Categorial Grammar: Logical Syntax, Semantics, and Processing*, Oxford University Press.

Morrill, G. & Fadda, M. (2008), 'Proof nets for basic discontinuous Lambek calculus', *Journal of Logic and Computation* **18**(2), 239–256.

Morrill, G., Valentín, O. & Fadda, M. (2011), 'The Displacement calculus', *Journal of Logic, Language and Information* **20**(1), 1–48.

Pentus, M. (2006), 'Lambek calculus is NP-complete', *Theoretical Computer Science* **357**(1), 186–201.

Valentín, O. (2014), The hidden structural rules of the discontinuous Lambek calculus, *in* Casadio et al. (2014), pp. 402–420.