Filtering Axiom Links for Proof Nets

Richard Moot

June 29, 2007

FG 2007:
The 12th conference on Formal Grammar
Dublin, Ireland
August 4-5, 2007

Organizing Committee:
Laura Kallmeyer **Paola Monachesi**
Gerald Penn **Giorgio Satta**

CENTER FOR THE STUDY
OF LANGUAGE
AND INFORMATION

June 29, 2007

1

Filtering Axiom Links for Proof Nets

Richard Moot

Abstract

An important problem for proving statements in multimodal categorial grammar is that even when using proof nets — which improve upon proof search in natural deduction and sequent calculus by identifying all equivalent proofs — the number of possible axiom links to consider is still enormous. We will propose several efficient strategies to reduce the number of axiom links and will evaluate the resulting combined strategy against a large number of random, provable Lambek calculus statements and find that we eliminate 97.92% of the planar axiom links which do not correspond to any proof net.

Keywords categorial grammar, proof nets, proof search, Lambek calculus

1.1 Introduction

The multimodal Lambek calculus (Moortgat, 1997) is a powerful and flexible grammar framework. Unfortunately, it has some sublogics — like the associative Lambek calculus L or the associative, commutative Lambek-Van Benthem calculus LP — which are known to be NP complete (Pentus, 2006, Kanovich, 1991).

In this article we will look at proof nets for the multimodal Lambek calculus and investigate ways to reduce the number of axiom links we need to perform in order to decide whether a statement $\Gamma \vdash C$ is derivable or not.

We will then evaluate the effect of these reductions on a large number of randomly generated Lambek calculus sequents and find that the combined filtering strategies filter our the large majority of the incorrect

FG-2007.
Organizing Committee:, Laura Kallmeyer, Paola Monachesi, Gerald Penn, Giorgio Satta.

axiom links, that is, those axiom links which are not part of any proof net, showing that — in spite of the theoretical complexity — in practice we can parse Lambek grammar while making just a small percentage of incorrect axiom links.

The rest of this paper is structured as follows. Section 1.2 will give a short introduction to parsing statements of the multimodal Lambek calculus using proof nets. Section 1.3 will discuss four strategies for reducing the total number of axioms links. The first two: acyclicity and connectedness (Section 1.3.1) and first-order approximation (Section 1.3.2) are fairly well-known and I will touch upon them only briefly. The final two strategies are new. We will show how to compute the relations between pairs of unary connectives using a context free grammar in Section 1.3.3 and how to eliminate axiom links which cannot be part of a *total* linking using methods from constraint logic programming in Section 1.3.4. Section 1.4 will evaluate the combined algorithm against a set of derivable Lambek calculus sequents.

1.2 Proof nets

We will follow Moot and Puite (2002) in our presentation of proof nets in this section. Constructing a proof structure for a statement in the multimodal Lambek calculus is done in three phases:

1. decompose the lexical formulas as well as the goal formula,
2. connect positive and negative atomic formulas,
3. decide whether the resulting proof structure is a proof net, using a correctness criterion.

We will look at each of the three phases in turn. Decomposing the lexical formulas is deterministic and linear in the number of connectives in the statement. Depending on whether a complex formula is a hypothesis (antecedent formula) or a conclusion (succedent formula), only one link can apply and we can simply descend the formula tree until we reach the leaves, which are the atomic formulas.

The full set of links is shown in Table 1. There are two links for every connective: one for when it occurs as a hypothesis (portrayed at the bottom; antecedents formulas start their unfolding here) and one for when it occurs as a conclusion (portrayed at the top; the goal formula starts its unfolding here).

The rule for $B\backslash_i A$ as a hypothesis is simply the modus ponens rule, saying it combines with a B on its left to form an A. The rule for $B\backslash_i A$ as a conclusion is its exact opposite: it allows us to use a B hypothesis to prove an A. Of course, we'll need to check that the B occurs as the

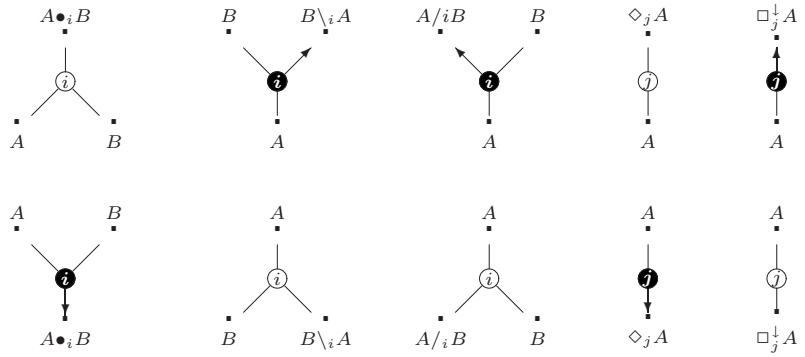

TABLE 1 Logical links for the multimodal Lambek calculus

leftmost sister of the A node and this is where the contractions will come in later.

Figure 1 shows an example unfolding of the sequent

$$np, (np\backslash s)/np, ((np\backslash s)/np)\backslash(np\backslash s) \vdash s$$

which would be a lexical lookup for a sentence like 'Robin wet himself'. The numbers on the atomic formulas are not a part of the proof structure, they serve only to help us refer to the different formulas later.

This is a slightly simplified version of the abstract proof structures of Moot and Puite (2002). We have suppressed as much information as possible: the information on the internal nodes, the lexical formula (both of them can be uniquely reconstructed from the unfolding) and the premiss/conclusion distinction of the individual nodes in Figure 1 as well (since we need it only when there is just a single antecedent formula).

The second part of constructing a proof net is identifying the axiomatic formulas: we select a positive and a negative atomic formula of the same type and identify their vertices, thereby connecting different parts of the graph. For this second step, there are potentially many solutions and the main part of this paper will focus on strategies for reducing these possibilities as much as possible.

Figure 1 shows the possibilities for the axioms links by portraying them in a grid: every row represents the axiom link possibilities for a negative formula, whereas every column represents the axiom link possibilities for a positive formula. A full axiom linking corresponds to putting exactly one mark in every row and column. In graph theory, this is usually called a *perfect matching*.

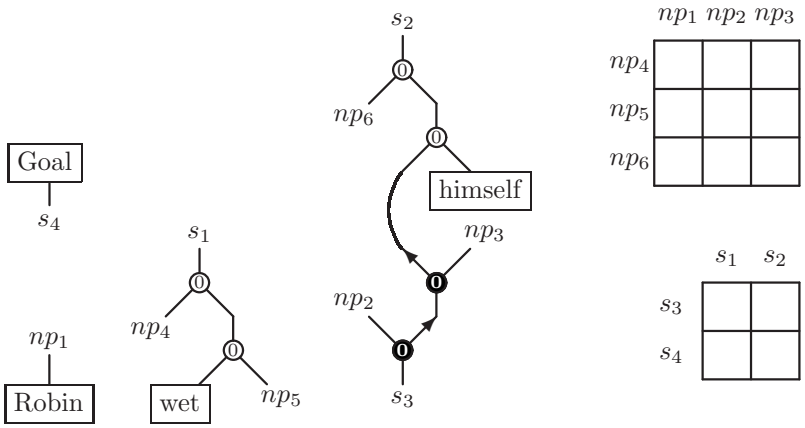

FIGURE 1 Formula unfolding and axiom link possibilities for $np, (np\backslash s)/np, ((np\backslash s)/np)\backslash(np\backslash s) \vdash s$

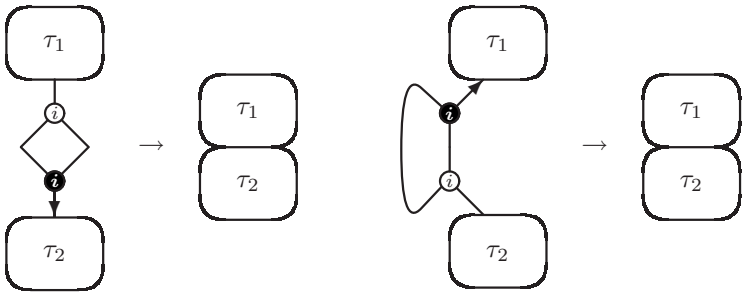

TABLE 2 Graph contractions

The final step is checking if the resulting graph is a proof net, by contracting all par links, drawn with the black center, as shown in Table 2. When all par links have been contracted, the result will be a tree with the lexical entries used as its leaves.

In case it is impossible to contract a par link, the proof structure we are dealing with is not a proof net and therefore, we know there is no proof of the corresponding sequent.

1.3 Filtering Axiom Links

Simple combinatorics shows that there are $n!$ possible axiom linkings for $2n$ atomic formulas. In Figure 1, there are therefore 2 possibilities for s and 6 possibilities for np. This makes exhaustive search prohibitive

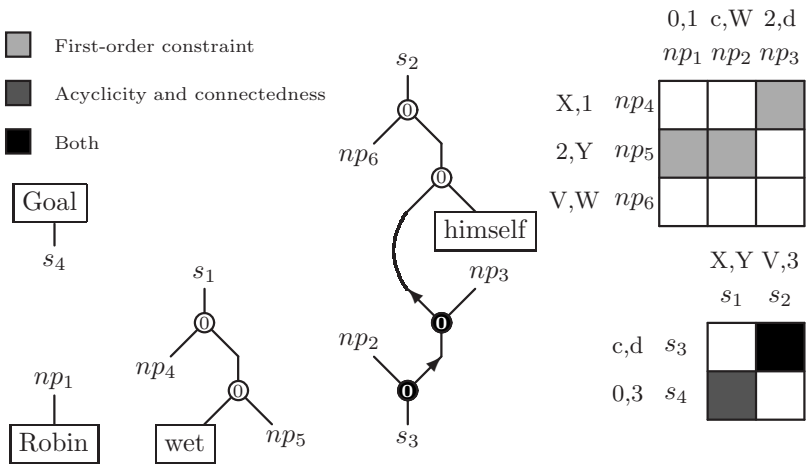

FIGURE 2 Links excluded by the acyclicity and connectedness condition

for all but the most trivial statements.

However, there are several possibilities to rule out axiom links which can never contribute to a contractible proof structure. We discuss some known and some new strategies in the following sections.

1.3.1 Acyclicity and Connectedness

Danos and Regnier (1989) introduce the acyclicity and connectedness criterion for proof nets of multiplicative linear logic. Give that the categorial logics we work with are all sublogics of multiplicative linear logic (in the sense that any derivable sequent in multimodal categorial grammar has a derivable image in multiplicative linear logic) we can use the fact that anything underivable in MLL is underivable in categorial grammar as well.

Moot (2004) shows how an adaptation of the Floyd-Warshall algorithm can be used to select from the total set of possible axiom links those that produce acyclic and connected proof structures. The complexity of this application of the Floyd-Warshall algorithm is $O(n^4)$.

Figure 2 shows, in dark gray and black, the links which are excluded when we use this condition. We remark that this leaves just one possibility for linking the s formulas.

1.3.2 First-order Approximation and Word Order

Even though the acyclicity and connectedness check is an effective test, it is based on the 'worst case' scenario of a fully associative and commutative logic. A second strategy for removing axiom links which cannot

contribute to constructing a proof net is to use first-order approximation to take constraints on word order into account. This has been used at least since LLoré and Morrill (1995) (though in a slightly different context).

Moot and Piazza (2001) propose an embedding of the Lambek calculus using first-order quantifiers and show how several linguistic phenomena like quantifier scope ambiguities, *wh* extraction and island constraints — for which there is no satisfactory treatment in the Lambek calculus — can be given an analysis using first-order quantifiers as well.

As long as we make sure that the structural rules permit a subset of the word order possibilities allowed by the first order variables, this strategy is correct.

Figure 2 shows the first-order labels assigned to the atomic formulas of our example sequent and — in light grey and black — the axiom links which are excluded using the first order constraints for L, which is justified given that we have a sequent which is derivable even in the non-associative Lambek calculus. In the figure, the numbers $0, 1, \ldots$ correspond to constants referring to string positions, lower-case letters $c, d, \ldots$ to constants introduced by the par links and upper-case letters $X, Y, \ldots$ correspond to variables.

However, we can assign slightly more subtle first-order variables and constants. An example for the treatment of extraction is shown in Figure 3. Here, we simply use the solution of Moot and Piazza (2001) for *wh* words. The first-order variables indicate that the word 'which', when it occurs between string positions 1 and 2, is looking for an n starting at some position X directly to its left and an s ending at position Y directly to its right. Inside this s we can use an np which can take up any position. The final result will then be an n between X (the leftmost position of the n argument) and Y (the rightmost position of the s argument). Note that we still need to add the appropriate structural rules to our grammar in order to derive medial extraction cases (see Moortgat, 1997, for a solution).

Given that we can check the first order constraints simply by unification of variables and constants, meaning $O(1)$ per cell in the axiom matrix, the total complexity of the first-order constraints is $O(n^2)$ for $2n$ atomic formulas.

1.3.3 The Unary Connectives

The unary connectives are a powerful addition to the multimodal Lambek calculus. They can be used to license structural rules but the relations between the logical formulas they induce can be used to encode linguistic features, as done, for example by Heylen (1999), or to restrict

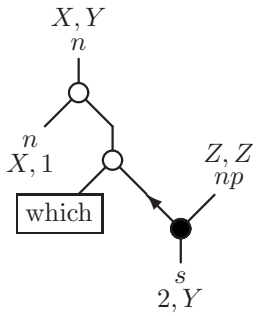

FIGURE 3 First order variables for 'which'

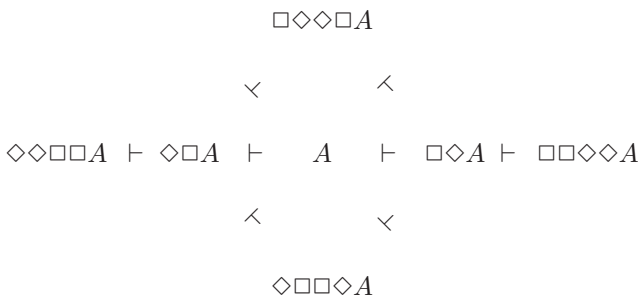

FIGURE 4 Derivability relations using the unary logical rules

scope possibilities, as done, for example by Bernardi and Moot (2003).

In certain cases, we use the unary modalities just for the derivability relations between the different types. Figure 4 summarizes the different relations between unary prefixes of up to four. Note that $\Diamond\Box\Diamond\Box A$ and $\Box\Diamond\Box\Diamond A$ are not displayed, given that they are equivalent to $\Diamond\Box A$ and $\Box\Diamond A$ respectively[1].

By extending the unary prefixes further we can generate an intricate hierarchy of formulas, but for many applications the seven formula recipes in Figure 4 suffice. However, even for the formulas we've shown here it's not directly clear which pairs of them are derivable.

Fortunately, there is a simply way to check the contractibility of a sequence of unary formulas. First, we convert this sequence to a string as follows (with ϵ being the empty string and . the concatenation operation)

Definition 1 Let A and B be two formulas which are the identical up

[1] Technically the graph of Figure 4 is the transitive reduction of the derivability relation where all equivalent formulas (which would correspond to cycles in the unreduced graph) have been replaced by their smallest element.

to their unary prefixes. $\sigma(A \vdash B)$, the *string corresponding to* $A \vdash B$, is defined as $\|A\|^-.\|B\|^+$, where the positive and negative formula are translated as follows.

$$\begin{array}{llll} \|\Diamond B\|^+ = & \|B\|^+.m & \|\Diamond A\|^- = & l.\|A\|^- \\ \|\Box B\|^+ = & \|B\|^+.r & \|\Box A\|^- = & \mathrm{m}.\|A\|^- \\ \|B\|^+ = & \epsilon \text{ otherwise} & \|A\|^- = & \epsilon \text{ otherwise} \end{array}$$

The easiest way to see the correspondence between a sequence of unary modes and a string is by turning a page with a formula unfolding 90 degrees to the right and realizing that every arrow pointing left will produce an l, while every arrow pointing right will produce an r.

Now, given that we have produced a string corresponding to the two sequences of unary connectives, we can check contractibility of these unary modes using the following context free grammar.

$$\begin{array}{lll} \mathrm{S} \rightarrow \epsilon & & (1) \\ \quad \mid \mathrm{l\ S\ m\ S} & & (2) \\ \quad \mid \mathrm{m\ S\ r\ S} & & (3) \end{array}$$

Rule (1) corresponds to the fact that it is possible not to have any unary connectives in front of a formula at all. Rule (2) corresponds to the $\Diamond$ contraction and rule (3) to the $\Box$ contraction.

Proposition 1 *$A \vdash B$ contracts to a single vertex using the unary contractions iff $S \rightarrow \sigma(A \vdash B)$.*

Proof (sketch) $\Rightarrow$ Induction on the number of contractions c. If $c = 0$ we use rule (1). If $c > 0$ there we look at the point where the *first* link is contracted. In order for this contraction to be valid the links between the two contracted links need to contract to a single node and in order for the entire sequence to contract everything after the second link needs to contract to a single node as well. Induction hypothesis allows us to combine these smaller proof nets using either rule (2) or (3).

$\Leftarrow$ Induction on the length of the CFG derivation. $\Box$

To give an illustration of how to use the context free grammar, we show how the derivable sequent $\Diamond\Box\Box\Diamond A \vdash \Box\Diamond A$ translates to *lmmlmr*, which we can derive as shown below.

$$\begin{array}{l} \mathrm{S} \rightarrow \mathrm{l\ S\ m\ S} \\ \quad \rightarrow \mathrm{l\ m\ S} \\ \quad \rightarrow \mathrm{l\ m\ m\ S\ r\ S} \\ \quad \rightarrow \mathrm{l\ m\ m\ S\ r} \\ \quad \rightarrow \mathrm{l\ m\ m\ l\ S\ m\ S\ r} \\ \quad \rightarrow \mathrm{l\ m\ m\ l\ m\ S\ r} \\ \quad \rightarrow \mathrm{l\ m\ m\ l\ m\ r} \end{array}$$

We can show the inverse statement $\Box\Diamond A \vdash \Diamond\Box\Box\Diamond A$ which would

correspond to *mlmrrm* is underivable simply because we cannot match the final m: there is no r to its right and if we would match it to the single l we would need to derive *mrr*, the symbols in between, but this is impossible given that it has an odd number of symbols.

The context free grammar is easily extended to the multimodal case, simply by adding different symbols l_i, m_i and r_i to the grammar and adding two new grammar rules for each of the new symbols.

There are limitations to using this system, however. First of all, it requires us to remove all unary branches from the final tree, though we could overcome this limitation by adding the following rule

$$\mathrm{S} \rightarrow \mathrm{m\ S}$$

to our grammar. Secondly, we cannot use this strategy if some of the structural rules for the unary modes we're interested in are incompatible with the formula to string translation. Examples would be any inclusion rules between unary modes (though adding grammar rules would again be an option here) or structural rules which move unary modes up or down the tree, like the K, K1 and K2 structural rules of Moortgat (1997), which would require changing the translation function.

Given that parsing a context free grammar is $O(n^3)$ and we would have to perform this calculation for all n^2 possible axiom links, the total $O(n^5)$ complexity is somewhat high. So for grammars which use the unary connectives extensively, it can be beneficial to pre-compute the relations between all sequences of unary connectives occurring in the grammar, after which we can do a simple table lookup to see if contraction is possible. This would reduce the total complexity to $O(n^2)$.

1.3.4 Régin's Algorithm

Even with all the previous constraints on axiom links in place, we sometimes fail to exclude some axiom links which cannot belong to a total matching. This is in part because an axiom link is regarded more or less in isolation, meaning that we don't exploit the fact the we need to find a *total* matching.

Régin (1994) proposes an algorithm for the slightly more general problem of finding solutions for 'all different' constraints in constraint logic programming. His algorithm separates the possible axiom links (in our case) into three categories: those which must be a part of any linking, those who are only part of some linkings and those which do not belong to any linking.

As an trivial example of Régin's algorithm, it would reduce a graph as shown in Figure 5 on the left (which we would obtain using the

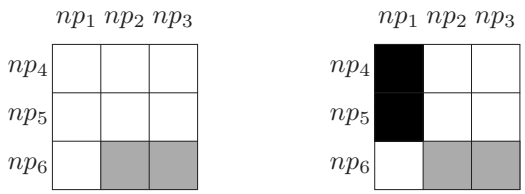

FIGURE 5 A trivial application of Régins algorithm

acyclicity and connectedness constraint after linking both s formulas of Figure 2) to the more logical structure on the right, simply because the unique solution left for np_6 would prevent np_1 from being used for any other axiom links.

Even though a smart axiom linking strategy (like always linking the atomic formula with the least possibilities first) would not benefit a lot from the reduction in this example, there are cases where it will exploit information like the absence of a total matching to fail directly.

The total complexity for Régin's algorithm is $O(s^2d^2)$ where s is the number of source nodes and d is the number of destination nodes. Given that in our application both of these are n (for $2n$ atomic formulas), the complexity will be $O(n^4)$ which means the total complexity for filtering all constraints is $O(n^4)$ as well.

1.4 Evaluation

In order to evaluate the combined filtering strategies, we have tested the axiom constraints on randomly generated derivable statements of the Lambek calculus. These statements have been generated using the inductive definition of Lambek calculus proof nets where all duplicate statements have been removed. To make the task as difficult as possible, only a single atomic formula a without unary prefixes has been used. Unfortunately, underivable statements don't have such an easy inductive characterization.

Figure 6 shows the amount of statements by the number of atomic formulas per sequent in the sample set, as well as the number of sequents by the number of connectives.

Out of 15.946 possible planar axiom links and 61.524 total possible axiom links, 2.546 correspond to different proofs. Therefore, there are 13.400 planar axioms links and 58.978 total axiom links which do not belong to any proof. Of these, the combined filtering algorithm excludes all but 279, for a total of 2.825 axiom links performed.

This means we eliminated 97.92% of the incorrect planar axiom links and 99.53% of the total number of incorrect axiom links. From the

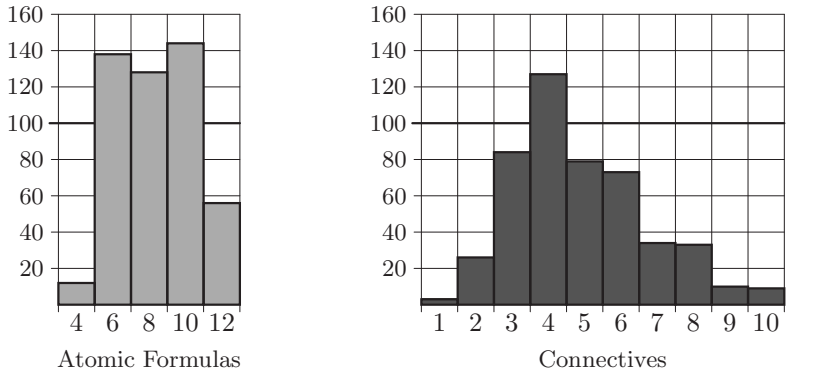

FIGURE 6 The distribution of the total number of atomic formulas and the total number of connectives in the randomly generated derivable statements

perspective of the *correct* axiom links, we perform only 10.96% more links than a 'perfect' linking strategy — which through some unknown method would only find links corresponding to proofs in an NP complete logic.

1.5 Conclusions and Future Work

We have seen that in spite of the computational complexity of computing all axiom links for a given statement, a combination of constraints on the possible axiom links can reduce the total number of axiom links to just a bit over the optimal number of axiom links to be performed.

Some important questions remain unresolved. For example, are there conditions where the filtered axiom links correspond *exactly* to the axiom links which belong to a proof net? This would mean that after the filtering algorithm has done its job, the resulting axiom matrix would contain all and only those links which would be used for a proof net. In that case, the axiom possibilities would form a sort of shared representation of all proofs for a statement and moreover, it would be computed in $O(n^4)$ time. However, analyzing the incorrect axiom links of our experiment doesn't seem to give an easily identifiable handle on the subclasses of multimodal categorial grammar which would have this property.

Another interesting line of research would be to evaluate the current algorithms against a large corpus of categorial grammar sentences, such as those of Hockenmaier (2003) and Moot (2007) and see how well the proposed linking strategy scales up to real-world applications.

References

Bernardi, Raffaella and Richard Moot. 2003. Generalized quantifiers in declarative and interrogative sentences. *Logic Journal of the IGPL* 11(4):419–434.

Danos, Vincent and Laurent Regnier. 1989. The structure of multiplicatives. *Archive for Mathematical Logic* 28:181–203.

Heylen, Dirk. 1999. *Types and Sorts: Resource Logic for Feature Checking*. Ph.D. thesis, Utrecht Institute of Linguistics OTS, Utrecht University.

Hockenmaier, Julia. 2003. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Kanovich, Max. 1991. The multiplicative fragment of linear logic is NP-complete. Tech. rep., University of Amsterdam. ITLI Prepublication Series X-91-13.

LLoré, F. Xavier and Glyn Morrill. 1995. Difference lists and difference bags for logic programming of categorial deduction. In *Proceedings of SEPLN XI*. Deusto.

Moortgat, Michael. 1997. Categorial type logics. In J. van Benthem and A. ter Meulen, eds., *Handbook of Logic and Language*, chap. 2, pages 93–177. Elsevier/MIT Press.

Moot, Richard. 2004. Graph algorithms for improving type-logical proof search. In *Proceedings Categorial Grammars 2004: an Efficient Tool for Natural Language Processing*. Elsevier.

Moot, Richard. 2007. Automated extraction of type-logical supertags from the spoken dutch corpus. In S. Bangalore and A. Joshi, eds., *Complexity of Lexical Descriptions and its Relevance to Natural Language Processing: A Supertagging Approach*. MIT Press. to appear.

Moot, Richard and Mario Piazza. 2001. Linguistic applications of first order multiplicative linear logic. *Journal of Logic, Language and Information* 10(2):211–232.

Moot, Richard and Quintijn Puite. 2002. Proof nets for the multimodal Lambek calculus. *Studia Logica* 71(3):415–442.

Pentus, Mati. 2006. Lambek calculus is NP-complete. *Theoretical Computer Science* 357(1–3):186–201.

Régin, Jean-Charles. 1994. A filtering algorithm for constraints of difference in CSPs. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 362–367. Seattle: AAAI.