Memoisation of categorial proof nets: parallelism in categorial processing[1]

Glyn Morrill
Dept. de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Pau Gargallo, 5
08028 Barcelona

`morrill@lsi.upc.es, http://www-lsi.upc.es/~glyn/`

21st March 1996

[1]Thanks to Ph. de Groote, J-M. Merenciano and C. Retoré for discussion relating to this work.

Abstract

We introduce a method of memoisation of categorial proof nets. Exploiting the planarity of non-commutative proof nets, and unifiability as a correctness criterion, parallelism is simulated through construction of a proof net matrix of most general unifiers for modules, in a manner analogous to the Cocke-Younger-Kasami algorithm for context free grammar.

Memoisation of categorial proof nets: parallelism in categorial processing

1 Introduction

If the evolutionary tendency of grammatical formalisms could be summed up in one word, that word could well be lexicalism. The lexicon was once considered the locus of all and only idiosyncratic information; it may be hard now to find any proponents at all of such a view. Rather, one hears of the balance or tradeoff between lexicon and syntax: the tenet that the lexicon should comprise only what is idiosyncratic is, simply, no longer held.

The notion that there is a compromise to be struck between lexicon and syntax is in turn rejected in the radical lexicalism of (type logical) categorial grammar (van Benthem 1991, Morrill 1994, Moortgat 1996). According to this view a grammar comprises *only* a lexicon.

The difference between syntactic and lexical encoding of grammatical information is typified by the contrast between (pure) context free grammar and Lambek categorial grammar. It is highlighted in particular by the proof that the class of string sets generated by context free grammars and that of those generated by the associative Lambek calculus are equal (Pentus 1993; the non-associative Lambek calculus also generates exactly the same class, see Buszkowski 1986).

The computational attraction of lexicalised information has the following intuitive rendering. Suppose we need to analyse some twenty word sentence; then we do so by accessing the (fully) lexicalised properties of those twenty words, and compute according to just that information. It makes no difference whether the grammar covers constructions ranging over a vocabulary of 200, 2,000, or 20,000 words: all that is relevant is lexically *localised* to those twenty words occurring and if, say, there are no wh-elements, coordinators, or quantifiers, then the complexities of those constructions are not suffered in the analysis.

In a context free grammar, by contrast, an order of magnitude increase in grammar coverage is accompanied by an order of magnitude increase in number of rules. And in analysing a sentence, all rules potentially apply: there is a *globalisation* of information. (Of course one could devise tests to restrict attention to rules relevant to the words occuring – but that would be precisely to set off down the path of lexicalisation.)

One may see categorial grammar as normalised at the expense of size of lexical categories (rule set size is constant) while one may see context free grammar as normalised at the expense of rule set size (category size – atomic – is constant). But despite the intuitive allusion above, while polynomial time recognition algorithms exist for context free grammar, no such result has been shown for associative Lambek calculus (for the non-associative case see Aarts and Trautwein 1995).

The polynomial methods for context free grammar are based on use of charts, tabularisation, or memoisation. The purpose of the present article is to establish such methods also for categorial grammar.

Earlier efforts in this direction are made in König (1994) and Hepple (1992). These take as their point of departure Cut-free sequent proof search. It is not necessary to enter into details in order to see why this approach encounters a profound problem. A chart, or table, as used for context free grammar is a global data structure on which the progression of all tracks of derivation over a fixed list of premises (words or categories) are recorded in parallel in such a way that converging routes of analysis do not lead to duplication of computation. But categorial sequent calculus does not operate over fixed premises: it is essentially dynamic, with premises both growing and shrinking in the course of inference. From a sequent perspective a single global chart of the kind used for context free grammar cannot suffice. Hence we find, for example, recursive emission of “minicharts” in König (1994).

In chart methods we seek (simulation of) parallelism. When one identifies the notion of parallelism in categorial grammar it is found not in sequent proof but in *proof nets*. Roorda (1991) develops the notion of proof net for Lambek calculus in a manner corresponding to its original introduction in Girard (1987) for linear logic. If proof nets represent parallelism in categorial grammar it is perhaps in relation to these that one should develop tabular methods. In fact, Roorda already established that while linear logic proof nets are unordered, categorial proof nets are ordered as a list (or actually, a loop), and that they have a planar (non-crossing) structure. At first appearances then categorial proof nets exhibit precisely the characteristics — order and planarity — on which context free grammar memoisation depends. The complication that arises concerns *correctness* of proof nets, but this paper aims to show that in their generality these appearances are not deceptive and that memoisation of categorial proof nets yields a simulation of parallelism in categorial processing.

The organisation of the paper is as follows. In section 2 we present the notion of categorial proof net as an analogue of context free parse tree. In section 3 we present in a generic form the categorial proof net memoisation algorithm. In section 4 we exemplify the non-associative Lambek calculus, and in section 5 the associative Lambek calculus.

2 Categorial proof nets

Consider the following context free grammar:

(1)
1 S → N VP
2 N → DET CN
3 VP → ADV VP
4 VP → VP ADV

The following two serial rewriting derivations, although distinct, define the same construction S out of DET CN VP ADV.

(2)
a. S → N VP → DET CN VP → DET CN TV N
b. S → N VP → N TV N → DET CN TV N

On the other hand, the two serial derivations in (3) provide distinct constructions from ADV VP ADV into VP.

(3)
a. VP → ADV VP → ADV VP ADV
b. VP → VP ADV → ADV VP ADV

These distinctions arise because when we see each rule as an operation mapping from its daughter categories to its mother category, the composition of such operations arranged by distinct serial derivations may or may not be equal. For both the derivations in (2) the construction, or mode of composition, is that represented by the following tree (the usual parse tree, inverted).

(4)
$$\cfrac{\cfrac{\mathrm{DET}\quad\mathrm{CN}}{\mathrm{N}}\,2\qquad\cfrac{\mathrm{TV}\quad\mathrm{N}}{\mathrm{VP}}\,3}{\mathrm{S}}\,1$$

For (3) on the other hand we have the following different constructions.

(5) a.
$$\cfrac{\cfrac{\mathrm{ADV}\quad\mathrm{VP}}{\mathrm{VP}}\,3\qquad\mathrm{ADV}}{\mathrm{VP}}\,4$$

b.
$$\cfrac{\mathrm{ADV}\qquad\cfrac{\mathrm{VP}\quad\mathrm{ADV}}{\mathrm{VP}}\,4}{\mathrm{VP}}\,3$$

We see that the parse trees represent what underlies the serial rewriting derivations: a *partial* ordering of the rule applications. In the case of (2), 2 and 3 follow 1, but are not ordered with respect to each other: those steps may be performed simultaneously. The serial rewriting derivations must choose a total ordering, but the parse trees abstract away from irrelevant structure and are graphs representing just the partial ordering of steps which correspond to equivalence classes of serial derivations.

Consider now categorial grammar. Sequent calculi for the implicational fragments of the non-associative Lambek calculus **NL** (antecedents binary bracketed) and the associative Lambek calculus **L** (antecedents unbracketed) are as follows.

(6) a. $A \Rightarrow A$ id $\qquad \dfrac{\Gamma \Rightarrow A \qquad \Delta(A) \Rightarrow B}{\Delta(\Gamma) \Rightarrow B}\mathrm{Cut}$

b. $\dfrac{\Gamma \Rightarrow A \qquad \Delta(B) \Rightarrow C}{\Delta([\Gamma, A\backslash B]) \Rightarrow C}\backslash\mathrm{L} \qquad \dfrac{[A, \Gamma] \Rightarrow B}{\Gamma \Rightarrow A\backslash B}\backslash\mathrm{R}$

c. $\dfrac{\Gamma \Rightarrow A \qquad \Delta(B) \Rightarrow C}{\Delta([B/A, \Gamma]) \Rightarrow C}/\mathrm{L} \qquad \dfrac{[\Gamma, A] \Rightarrow B}{\Gamma \Rightarrow B/A}/\mathrm{R}$

(7) a. $A \Rightarrow A$ id $\qquad \dfrac{\Gamma \Rightarrow A \qquad \Delta(A) \Rightarrow B}{\Delta(\Gamma) \Rightarrow B}\mathrm{Cut}$

b. $\dfrac{\Gamma \Rightarrow A \qquad \Delta(B) \Rightarrow C}{\Delta(\Gamma, A\backslash B) \Rightarrow C}\backslash\mathrm{L} \qquad \dfrac{A, \Gamma \Rightarrow B}{\Gamma \Rightarrow A\backslash B}\backslash\mathrm{R}$

c. $\dfrac{\Gamma \Rightarrow A \qquad \Delta(B) \Rightarrow C}{\Delta(B/A, \Gamma) \Rightarrow C}/\mathrm{L} \qquad \dfrac{\Gamma, A \Rightarrow B}{\Gamma \Rightarrow B/A}/\mathrm{R}$

Cut-elimination — that every proof has a Cut-free normal form — yields a decision procedure because each logical inference rule has one less connective occurrence in its premisses than in its conclusion. However, distinct Cut-free sequent proofs of the same sequent may or may not define the same mode of composition. Consider the following two proofs of S/S, S, S\S $\Rightarrow$ S in **L**.

(8) a. $$\dfrac{\dfrac{\mathrm{S} \Rightarrow \mathrm{S} \quad \mathrm{S} \Rightarrow \mathrm{S}}{\mathrm{S/S, S} \Rightarrow \mathrm{S}}/\mathrm{L} \quad \mathrm{S} \Rightarrow \mathrm{S}}{\mathrm{S/S, S, S\backslash S} \Rightarrow \mathrm{S}}\backslash\mathrm{L}$$

b. $$\dfrac{\mathrm{S} \Rightarrow \mathrm{S} \quad \dfrac{\mathrm{S} \Rightarrow \mathrm{S} \quad \mathrm{S} \Rightarrow \mathrm{S}}{\mathrm{S, S\backslash S} \Rightarrow \mathrm{S}}\backslash\mathrm{L}}{\mathrm{S/S, S, S\backslash S} \Rightarrow \mathrm{S}}/\mathrm{L}$$

Under the usual Curry-Howard interpretation these assign distinct constructions, x: S/S, y: S, z: S\S $\Rightarrow$ $(z\ (x\ y))$: S and $(x\ (z\ y))$: S respectively. But consider now the following (this time in **NL**):

(9) a. $$\dfrac{\dfrac{\mathrm{CN} \Rightarrow \mathrm{CN} \quad \mathrm{N} \Rightarrow \mathrm{N}}{[\mathrm{N/CN, CN}] \Rightarrow \mathrm{N}}/\mathrm{L} \quad \mathrm{S} \Rightarrow \mathrm{S}}{[[\mathrm{N/CN, CN}], \mathrm{N\backslash S}] \Rightarrow \mathrm{S}}\backslash\mathrm{L}$$

b. $$\dfrac{\mathrm{CN} \Rightarrow \mathrm{CN} \quad \dfrac{\mathrm{N} \Rightarrow \mathrm{N} \quad \mathrm{S} \Rightarrow \mathrm{S}}{[\mathrm{N, N\backslash S}] \Rightarrow \mathrm{S}}\backslash\mathrm{L}}{[[\mathrm{N/CN, CN}], \mathrm{N\backslash S}] \Rightarrow \mathrm{S}}/\mathrm{L}$$

Both of these derivations generate the construction (10).

(10) $[[x\text{: N/CN}, y\text{: CN}], z\text{: N\backslash S}] \Rightarrow (z\ (x\ y))\text{: S}$

And given the sequent [N/CN, CN] $\Rightarrow$ S/(N\S) one may start with a right inference, or a left inference:

(11)a. $$\dfrac{[[\mathrm{N/CN, CN}], \mathrm{N\backslash S}] \Rightarrow \mathrm{S}}{[\mathrm{N/CN, CN}] \Rightarrow \mathrm{S/(N\backslash S)}}/\mathrm{R}$$

b. $$\dfrac{\mathrm{CN} \Rightarrow \mathrm{CN} \quad \mathrm{N} \Rightarrow \mathrm{S/(N\backslash S)}}{[\mathrm{N/CN, CN}] \Rightarrow \mathrm{S/(N\backslash S)}}/\mathrm{L}$$

Route (11a) reduces to (9) and (11b) is also successful: a total of three derivations yielding (12).

(12) $[x\text{: N/CN}, y\text{: CN}] \Rightarrow \lambda z(z\ (x\ y))\text{: S/(N\backslash S)}$

Since **NL** is a restriction of **L** (regulated according to the brackets), **L** itself is also subject to all these equivalences. The problem is that some, but not all, inference steps are permutable. To prove (12) we need /L, \L, and /R, but \L must follow /R, hence the three possibilities.

Proof nets are to this problem what parse trees are to the corresponding problem in context free grammar. They are graph structures representing equivalence classes of sequent proofs: not just normal forms subject to a particular ordering of steps, but structures embodying the partial ordering on inference steps in equivalence classes of sequential proofs.

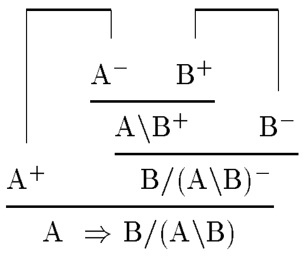
We consider the presentation of proof nets for Lambek calculus of Roorda (1991). Proof nets are built not over formulas but over the construction trees of formulas: trees with atoms at their leaves, and in which each mother node indicates the combination of its daughter subformulas by some connective. Thus the parts of a formula are laid out ("unfolded") for potential simultaneous access, not restricted according to the recursive nesting of connectives. Furthermore, whether items appear in antecedent or succedent positions is not indicated by location in a sequent, but by a marking of polarity. Our formulas are signed positive for antecedent occurrences and negative for succedent occurrences, and recursively unfolded as follows (our polarities are reversed with respect to Roorda: we see proof from the perspective of refutation).

(13) $$\dfrac{B^+ \quad A^-}{B/A^+} \qquad \dfrac{A^- \quad B^+}{A\backslash B^+} \qquad \dfrac{A^+ \quad B^-}{B/A^-} \qquad \dfrac{B^- \quad A^+}{A\backslash B^-}$$

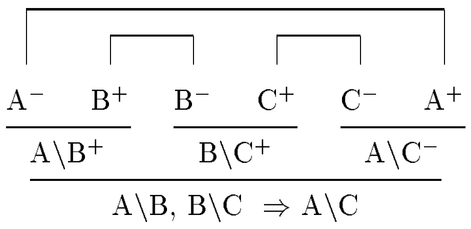
The transmission of polarities can be understood when we see an implication as a disjunction of its consequent with the negation of its antecedent. The steps given are compilations of decomposition accordingly, with unfolding, involution of negation, and De Morgan laws for conjunction or disjunction. The result of unfolding in this way the formulas in a sequent is called a *proof frame*. A *proof net* is made by connecting pairs of leaves with the same atoms and complementary polarities. The ordering given, which swaps the components of negative (i.e. succedent) occurrences of implications allows restriction to planar (that is, non-crossing) connections.

The following, for example, are proof nets for lifting A $\Rightarrow$ B/(A\B) and composition A\B, B\C $\Rightarrow$ A\C in **L**.

(14)



(15)



A linking of all literals is called a *proof structure*, but not all proof structures correspond to proofs, i.e. they are not all proof nets. Just by considerations of symmetry with respect to lifting we can see that there will be a proof structure for the invalid "lowering": B/(A\B) $\Rightarrow$ A. A *long trip condition* (Girard 1987) can express the required correctness condition in geometric terms. Here we use methods in which the required constraint is reduced to solvability of a unification problem.

Roorda (1991) and Moortgat (1992) present unfolding with prosodic labelling as follows.

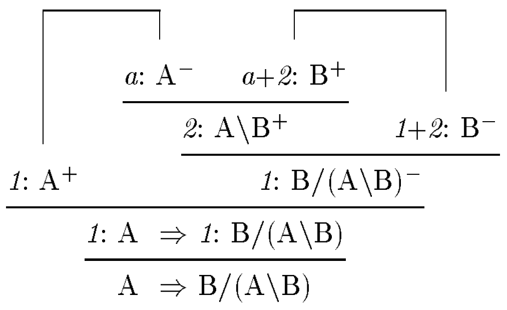
(16) a.
$$\frac{\gamma{+}a\colon B^+ \quad a\colon A^-}{\gamma\colon B/A^+} \qquad \frac{a\colon A^- \quad a{+}\gamma\colon B^+}{\gamma\colon A\backslash B^+} \qquad a \text{ new variable}$$

b.
$$\frac{k\colon A^+ \quad \gamma{+}k\colon B^-}{\gamma\colon B/A^-} \qquad \frac{k{+}\gamma\colon B^- \quad k\colon A^+}{\gamma\colon A\backslash B^-} \qquad k \text{ new constant}$$

The succedent (negative) unfoldings introduce (Skolem-like) constants; the antecedent (positive) unfoldings introduce variables. Linking identifies the labels of linked atoms and the correctness condition is that a proof structure is a proof net if and only if the set of term pairs induced by linking are unifiable.

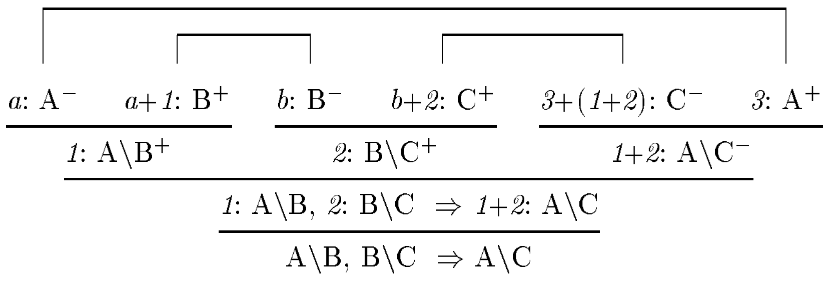
Consider lifting, for which there is the proof net (17).

(17)



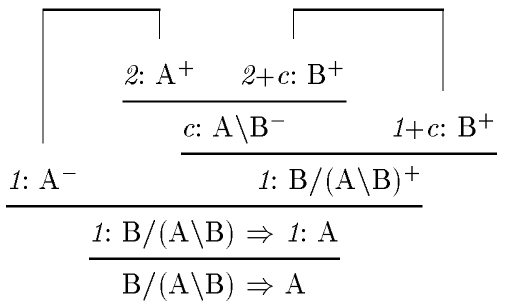
The linking yields the unification problem $\{1 = a, a{+}2 = 1{+}2\}$ which is clearly solved by the unifier $\{a = 1\}$. For composition we obtain (18).

(18)



This yields the unification problem $\{a = 3, a{+}1 = b, b{+}2 = 3{+}(1{+}2)\}$ which has the unifier $\{a = 3, b = 3{+}1\}$ in the associative case, but not in the non-associative one. For the invalid lowering however we have:

(19)



The unification problem $\{1 = 2, 1{+}c = 2{+}c\}$ clearly has no solution because 1 and 2 are distinct constants. Note incidently that in (19), antecedent and succedent are cycled at the moment of introducing polarities. Such cycles preserve the possibilities of constructing planar proof nets, i.e. the lists of polar formulas should be seen as forming a loop. We shall favour succedent-to-the-left to allow early top-down influence in left-to-right incrementation.

The method is attractive because it can be adapted to different calculi by unifying prosodic terms according to the laws, associativity and so forth, of the different algebras of interpretation. A partial linking of leaves defines a unification problem. If there is no unifier, no extension of this partial linking will be unifiable either and so the partial linking cannot form part of a proof net. If there is a unifier we have a partially constructed proof net, or what is called a *module*: a proof frame with some links made without violating the proof net correctness conditions. Our proposal is to tabularise modules connecting a continuous subsegment of leaves. For each, most general unifiers define the constraints on its further extension. Finally, the only elements of the unifier which are relevant to possible extensions are its assignments to variables reappearing outside the continuous subsegment covered; when we are just asking whether there exists some proof net validating a sequent, the value constraints on variables all the occurrences of which are within the subsegment do not matter regarding possible extensions.

The tabularisation process will resemble that of Cocke-Younger-Kasami (see Aho and Ullman 1972) for context free grammar in Chomsky normal form. Each subsegment is represented by a cell in a triangular matrix. In the CYK algorithm each cell entry is the set of nonterminal categories to which the subsegment belongs. Here, it is the set of unifiers constraining possible further extensions of alternative internal linkings.

We use the following grammar of planar linking:

(20) a. $M \rightarrow A\ \overline{A}$
b. $M \rightarrow A\ M\ \overline{A}$
c. $M \rightarrow M\ M$

We have only one non-terminal category, M, for continuous linkings; A and $\overline{A}$ indicate complementary terminal literals: ones with the same atom, but opposing polarity.

The proof net matrices computed by our algorithm are filled as shown in (21). The main diagonal is occupied by the ordered leaf literal assignments $\sigma_1\colon L_1, \ldots, \sigma_n\colon L_n$ resulting from unfolding of the sequent to be analysed. A cell entry $T(i, j)$ $(i<j)$ is the set of constraints (unifiers) that different linkings of the subsegment $\sigma_i\colon L_i, \ldots, \sigma_j\colon L_j$ can impose. Since linkings are binary there are only entries for even length subsegments.

(21)

	1	2	3	4	…	n
1	$\sigma_1\colon L_1$					
2	1st	$\sigma_2\colon L_2$				
3	X	2nd	$\sigma_3\colon L_3$			
4	4th	X	3rd	$\sigma_4\colon L_4$		
⋮	⋮	⋮	⋮	⋮	⋱	
6	last	X	penult.	X	…	$\sigma_n\colon L_n$

The algorithm is given in (22). It assumes the relation $MGU(\theta, \theta')$ in which θ and θ' are sets of term-term pairs (θ' will actually be a substitution, that is a set of variable-term pairs), and which states that θ' is a most general unifier solving the unification problem θ. Since we will deal with term unification, there is a unique most general unifier when one exists so MGU is a partial function from its first parameter to its second. The notation $\theta{\uparrow}_{i,k}$ indicates the restriction of the substitution θ according to those variables having occurrences outside $\sigma_i, \ldots, \sigma_k$, that is the restriction of θ to those variables which have occurrences outside the subsegment θ covers. The comments following "%" indicate the three rules applied at their relevant points. The main loop is for filling in successive rows. Starting at the right of a new row, (20a) is used to try to construct a module connecting two adjacent literals. The next loop completes the row from right to left; (20b) may build a module by connecting two literals separated by a module; (20c) may do so by combining any two adjacent modules covering a subsegment – hence the third loop.

(22)
```
for $k := 2$ to $n$ do
    begin
    $T(k{-}1, k) := \{\theta\uparrow_{k-1,k} | \sigma\colon L_{k-1}$ & $\sigma'\colon L_k$ are complementary,
                      and $MGU(\{\sigma = \sigma'\}, \theta)\}$; % $M \to A\ \overline{A}$
    for $i := k{-}3$ downto 1 in steps of 2 do
        begin
        $T(i, k) := \{\theta\uparrow_{i,k} | \sigma\colon L_i$ & $\sigma'\colon L_k$ are complementary, $\theta' \in T(j{+}1, k{-}1)$,
                      and $MGU(\theta' \cup \{\sigma = \sigma'\}, \theta)\}$; % $M \to A\ M\ \overline{A}$
        for $j := i{+}1$ to $k{-}1$ in steps of 2 do
            $T(i, k) := T(i, k) \cup \{\theta\uparrow_{i,k} | \theta' \in T(i, j), \theta'' \in T(j{+}1, k)$,
                      and $MGU(\theta' \cup \theta'', \theta)\}$; % $M \to M\ M$
        end
    end.
```

In the following two sections we show how this scheme can be applied in the case of the non-associative and associative calculi.

3 Non-associative Lambek calculus

Our first example is the sequent S, S\S ⇒ (S/S)\((N/S)\N). In the unfolding in (23) the succedent formula appears on the left. This is possible because of the cyclic invariance of proof nets. It allows a top-down influence to be brought to bear early in an incremental (left-to-right) analysis. Note that in the case of a sequent with more than two premisses in which a bracketing structure is not assumed, the succedent label would be left as a variable.

(23)
$$\dfrac{\dfrac{\dfrac{\mathbf{d}{+}(\mathbf{c}{+}(\mathbf{a}{+}\mathbf{b}))\colon \mathrm{N}^- \quad \dfrac{\mathbf{d}{+}e\colon \mathrm{N}^+ \quad e\colon \mathrm{S}^-}{\mathbf{d}\colon \mathrm{N/S}^+}}{\mathbf{c}{+}(\mathbf{a}{+}\mathbf{b})\colon \mathrm{(N/S)\backslash N}^-} \quad \dfrac{\mathbf{c}{+}f\colon \mathrm{S}^+ \quad f\colon \mathrm{S}^-}{\mathbf{c}\colon \mathrm{S/S}^+}}{\mathbf{a}{+}\mathbf{b}\colon \mathrm{(S/S)\backslash((N/S)\backslash N)}^-} \quad \mathbf{a}\colon \mathrm{S}^+ \quad \dfrac{g\colon \mathrm{S}^- \quad g{+}\mathbf{b}\colon \mathrm{S}^+}{\mathbf{b}\colon \mathrm{S\backslash S}^+}}{\mathrm{S,\ S\backslash S \Rightarrow (S/S)\backslash((N/S)\backslash N)}}$$

The completed matrix is as follows.

(24)

	1	2	3	4	5	6	7	8
1	$\mathbf{d}{+}(\mathbf{c}{+}(\mathbf{a}{+}\mathbf{b}))$: N$^-$							
2	$\{[e = \mathbf{c}{+}(\mathbf{a}{+}\mathbf{b})]\}$	$\mathbf{d}{+}e$: N$^+$						
3	X	{}	e: S$^-$					
4	$\{[f = \mathbf{a}{+}\mathbf{b}]\}$	X	$\{[e = \mathbf{c}{+}f]\}$	$\mathbf{c}{+}f$: S$^+$				
5	X	{}	X	{}	f: S$^-$			
6	{}	X	$\{[e = \mathbf{c}{+}\mathbf{a}]\}$	X	$\{[f = \mathbf{a}]\}$	$\mathbf{a}$: S$^+$		
7	X	{}	X	$\{[g = \mathbf{c}{+}\mathbf{a}]\}$	X	$\{[g = \mathbf{a}]\}$	g: S$^-$	
8	{[]}	X	$\{[e = (\mathbf{c}{+}\mathbf{a}){+}\mathbf{b}$, $e = \mathbf{c}{+}(\mathbf{a}{+}\mathbf{b})]\}$	X	$\{f = \mathbf{a}{+}\mathbf{b}\}$	X	{}	$g{+}\mathbf{b}$: S$^+$

We will not comment explicitly on every step in the construction of proof net matrices, but we will mention some which illustrate significant aspects of the algorithm.

In filling $T(3,6)$ the substitutions of $T(3,4)$ and $T(5,6)$ are combined. The unification problem presented is $\{e = \mathbf{c}+f, f = \mathbf{a}\}$ and this is solved with MGU $[e = \mathbf{c}+\mathbf{a}, f = \mathbf{a}]$. This unifier is then restricted to just e, because f has no occurrence outside the literal sequence $\sigma_3\colon L_3, \ldots, \sigma_6\colon L_6$.

Likewise, $T(1,4)$ is obtained by solving the unification problem $\{e = \mathbf{c}+(\mathbf{a}+\mathbf{b}), e = \mathbf{c}+f\}$. The MGU $[e = \mathbf{c}+(\mathbf{a}+\mathbf{b}), f = \mathbf{a}+\mathbf{b}]$ is restricted to f because e does not appear beyond $\sigma_1, \ldots, \sigma_4$.

Similarly, when $T(4,7)$ is filled the unifier $[g = \mathbf{c}+f]$ of σ_4 and σ_7 is combined with the substitution $[f = \mathbf{a}]$ of $T(5,6)$. The MGU $[g = \mathbf{c}+\mathbf{a}, f = \mathbf{a}]$ is then restricted to g.

$T(5,8)$ is filled by combining the unifier $[f = g+\mathbf{b}]$ of σ_5 and σ_8 with $[g = \mathbf{a}]$ in $T(6,7)$ and restricting to f.

The cell $T(3,8)$ can be occupied in one way by combining the unifier $[e = g+\mathbf{b}]$ of σ_3 and σ_8 with $[g = \mathbf{c}+\mathbf{a}]$ in $T(4,7)$. The result of this, restricted to e, is $\{[e = (\mathbf{c}+\mathbf{a})+\mathbf{b}]\}$. However the cell is also occupied by the result of combining $T(3,4)$ and $T(5,8)$. This yields the distinct unifier $\{[e = \mathbf{c}+(\mathbf{a}+\mathbf{b})]\}$, corresponding to that fact that it is possible to construct two distinct modules connecting $L_3, \ldots, L_8$, both of which are legitimate proof nets under construction. Were they to be completed, these proof nets would represent distinct modes of composition.

In filling $T(1,8)$ one may combine the unifier in $T(1,2)$ with the second of those in $T(3,8)$: they are the same and, of course, after restriction yield the empty substitution. But one may also combine the identical contents of $T(1,4)$ and $T(5,8)$, yielding again the empty substitution after restriction. We see then that only one of the modules for $T(3,8)$ actually gave rise to a proof net, and only one mode of composition of the sequent exists in **NL**. That the empty substitution was found twice in $T(1,8)$ originates in the fact that the grammar of planar linking is not unambiguous: *MMM* may be equivalently analysed to the left or to the right by (20c) because of the associativity of unification.

Consider the sequent (25) in relation to **NL**.

(25) R/(S/N), N ⇒ R/((N\S)/N)

The partial unfolding is as follows.

$$\frac{\dfrac{\mathbf{c}\colon (\mathrm{N}\backslash\mathrm{S})/\mathrm{N}^{+} \qquad (\mathbf{a}+\mathbf{b})+\mathbf{c}\colon \mathrm{R}^{-}}{\mathbf{a}+\mathbf{b}\colon \mathrm{R}/((\mathrm{N}\backslash\mathrm{S})/\mathrm{N})^{-}} \qquad \dfrac{\mathbf{a}+f\colon \mathrm{R}^{+} \qquad f\colon \mathrm{S}/\mathrm{N}^{-}}{\mathbf{a}\colon \mathrm{R}/(\mathrm{S}/\mathrm{N})^{+}} \qquad \mathbf{b}\colon \mathrm{N}^{+}}{\mathrm{R}/(\mathrm{S}/\mathrm{N}),\ \mathrm{N} \Rightarrow \mathrm{R}/((\mathrm{N}\backslash\mathrm{S})/\mathrm{N})} \tag{26}$$

It can already be seen that the connection of the two complementary R literals will fail, because their terms cannot be unified (in the absence of associativity). Thus (25) is shown not to be a theorem of **NL**.

4 Associative Lambek calculus

In the case of **L** the unification test for proof nets is for unification under associativity. This presents a problem for our method because terms do not have unique most general unifiers under associativity; for example, $a+b = \mathbf{c}+\mathbf{d}+\mathbf{e}$ has MGUs $[a = \mathbf{c}+\mathbf{d}, b = \mathbf{e}]$ and $[a = \mathbf{c}, b = \mathbf{d}+\mathbf{e}]$. Rather than try to work with labelling which lacks most general unifiers, we conjecture here a form of associativity-free unification check for **L**.

Labels for **L** will comprise two terms, representing the starting and ending positions of the sequence covered. We write $i - j$: A; thus:

(27)
$$\frac{0 - n\colon A^- \qquad 0 - 1\colon {A_1}^+ \qquad \dots \qquad n{-}1 - n\colon {A_n}^+}{A_1, \dots, A_n \Rightarrow A}$$

In the unfolding, variables and Skolem-like terms are introduced according to polarity; lists of zero or more variables (represented $\overline{v}$) are transmitted in the course of unfolding and used to construct dependent names.

(28) a.
$$\frac{i - k\colon B^{+,\overline{v},k} \qquad j - k\colon A^{-,\overline{v},k}}{i - j\colon B/A^{+,\overline{v}}} \qquad\qquad \frac{i - j\colon A^{-,\overline{v},i} \qquad i - k\colon B^{+,\overline{v},i}}{j - k\colon A\backslash B^{+,\overline{v}}}$$

b.
$$\frac{j - k(\overline{v})\colon A^{+,\overline{v}} \qquad i - k(\overline{v})\colon B^{-,\overline{v}}}{i - j\colon B/A^{-,\overline{v}}} \qquad\qquad \frac{i(\overline{v}) - k\colon B^{-,\overline{v}} \qquad i(\overline{v}) - j\colon A^{+,\overline{v}}}{j - k\colon A\backslash B^{-,\overline{v}}}$$

In (28a) the parameter (k or i) introduced reading up the page is a new variable; in (28b) it is a Skolem term with a new function letter.

First, we consider again the sequent S, S\S ⇒ (S/S)\((N/S)\N). Its unfolding in **L** is the following.

(29)
$$\frac{\dfrac{\dfrac{4 - 2\colon \mathrm{N}^{-,i} \qquad \dfrac{4 - i\colon \mathrm{N}^{+,i} \qquad 3 - i\colon \mathrm{S}^{-,i}}{4 - 3\colon \mathrm{N/S}^{+}}}{3 - 2\colon (\mathrm{N/S})\backslash\mathrm{N}^{-}} \qquad \dfrac{3 - j\colon \mathrm{S}^{+,j} \qquad 0 - j\colon \mathrm{S}^{-,i}}{3 - 0\colon \mathrm{S/S}^{+}}}{0 - 2\colon (\mathrm{S/S})\backslash((\mathrm{N/S})\backslash\mathrm{N})^{-}} \qquad 0 - 1\colon \mathrm{S}^{+} \qquad \dfrac{k - 1\colon \mathrm{S}^{-,k} \qquad k - 2\colon \mathrm{S}^{+,k}}{1 - 2\colon \mathrm{S}\backslash\mathrm{S}^{+}}}{\mathrm{S}, \mathrm{S}\backslash\mathrm{S} \Rightarrow (\mathrm{S/S})\backslash((\mathrm{N/S})\backslash\mathrm{N})}$$

The proof net matrix is (30).

(30)

	1	2	3	4	5	6	7	8
1	$4 - 2$: N$^-$							
2	{[$i = 2$]}	$4 - i$: N$^+$						
3	X	{}	$3 - i$: S$^-$					
4	{[$j = 2$]}	X	{[$i = j$]}	$3 - j$: S$^+$				
5	X	{}	X	{}	$0 - j$: S$^-$			
6	{}	X	{[$i = 1$]}	X	{[$j = 1$]}	$0 - 1$: S$^+$		
7	X	{}	X	{[$k = 3$]}	X	{[$k = 0$]}	$k - 1$: S$^-$	
8	{[]}	X	{[$i = 2$]}	X	{[$j = 2$]}	X	{}	$k - 2$: S$^+$

When $T(1,4)$ is filled on the basis of $T(1,2)$ and $T(3,4)$, the unifier $[i = 2, j = 2]$ is restricted to just j. Similarly, when $T(3,6)$ is filled on the basis of $T(3,4)$ and $T(5,6)$, the unifier $[i = 1, j = 1]$ is restricted to just i. $T(4,7)$ is filled on the basis of L_4 and L_7, and $T(5,6)$, and the j-constraint forgotten. Likewise, when $T(5,8)$ is filled on the basis of L_5 and L_8, and $T(6,7)$, the k-constraint is forgotten. One member of $T(3,8)$ is the unifier $[k = 3, i = 2]$, restricted to i, obtained from L_3 and L_8, and $T(4,7)$. When we combine $T(3,4)$ and $T(5,8)$ we get the unifier $[i = 2, j = 2]$ which, restricted to i, is also $[i = 2]$. These sources correspond to *different* modules linking $L_3, \dots, L_8$ (corresponding to different modes of semantic composition), which nevertheless impose exactly the same constraints on the remaining possible linkings of their context. On the other hand, when $T(1,8)$ is filled, through $T(1,2)$ and $T(3,8)$, or through $T(1,4)$ and $T(5,8)$, the alternatives arise through the non-semantically significant ambiguity of the grammar's analysis of MMM (over $L_1 - L_2, L_3 - L_4, L_5 - L_8$. Just as in the CYK recognition algorithm for CFG, alternatives which may or may not be semantically relevant are compacted through the unioning into cell entries of the external constraints imposed, not the full internal histories. Also just as for CYK, these internal histories can be recorded in a recoverable fashion by marking each category (in out case, unifier) with a set of pointers to its immediate generators.

Second, we consider again the sequent R/(S/N), N $\Rightarrow$ R/((N\S)/N). The unfolding in **L** is (31).

(31)

$$\dfrac{\dfrac{\dfrac{\dfrac{j-2\text{: N}^{-,i,j} \quad j-i\text{: S}^{+,i,j}}{2-i\text{: N\\S}^{+,i}} \quad 3-i\text{: N}^{-,i}}{2-3\text{: (N\\S)/N}^{+}} \quad 0-3\text{: R}^{-}}{0-2\text{: R/((N\\S)/N)}^{-}} \quad \dfrac{0-k\text{: R}^{+,k} \quad \dfrac{k-4(k)\text{: N}^{+,k} \quad 1-4(k)\text{: S}^{-,k}}{1-k\text{: S/N}^{-,k}}}{0-1\text{: R/(S/N)}^{+}} \quad 1-2\text{: N}^{+}}{\text{R/(S/N), N} \Rightarrow \text{R/((N\\S)/N)}}$$

In the matrix (32), $T(2,7)$ is filled from L_2 and L_7, and $T(3,6)$; the unifier $[k = 3, i = 4(3), j = 1]$ is restricted to j.

(32)

	1	2	3	4	5	6	7	8
1	j – 2: N$^-$							
2	{}	j – i: S$^+$						
3	X	{}	3 – i: N$^-$					
4	{}	X	{}	0 – 3: R$^-$				
5	X	{}	X	{[$k = 3$]}	0 – k: R$^+$			
6	{}	X	{$k = 3$, $i = 4(3)$}	X	{}	k – 4(k): N$^+$		
7	X	{[$j = 1$]}	X	{}	X	{}	1 – 4(k): S$^-$	
8	{[]}	X	{}	X	{}	X	{}	1 – 2: N$^+$

Thus the sequent is shown to be valid in **L**.

5 Conclusion

Independently of the details of labelled unfolding, the content of the present proposal should be clear enough. It is to use unifiability tests to check the correctness of categorial proof nets, and to memoise categorial deductions by tabularising the MGUs for modules constituting partial proof nets. We conclude with consideration of some matters and possibilities this raises.

One question is whether the current algorithm for **L** could be shown to determine **L**-validity in polynomial time. Of course **L** may simply not be polynomially decidable. Our concern here has been with the introduction of categorial proof net tabularisation as a method, rather than a means to answer a specific open problem; nevertheless, if **L**-validity can be decided in polynomial time, it presumably must be through a tabularisation of modules such as that here.

We have considered the Lambek calculi, but these pure systems suffer the same order of inadequacy as CFG itself regarding natural language grammar. Typically, natural grammar exercises crossed dependencies whereas these formalisms allow just nested dependencies. A central question then is how the approach might be extended to accommodate generalisations of Lambek calculi, particularly those allowing partial commutativity: can some approximation to planarity be preserved? A list of n literals has only $\frac{n^2}{2}$ continuous sublists, but a bag of n literals has 2^n subbags, so tabularisation within a space assuming free commutativity seems impracticable.

The algorithm given works on sequents, but in grammar one works with potentially ambiguous vocabulary. If one factors out sequent validation from lexical insertion, a string of n words each, say, ambiguous between two lexical categories demands solution of 2^n sequent validation subproblems. Lexical insertion and lexical ambiguity therefore must be accommodated in the tabularisation process. One also requires integration of semantic composition in such a way that readings are extractable from proof net matrices.

Finally, we suggest tentatively that proof net matrices may generate some wider prospects for categorial grammar. In the area of *robust* parsing one is interested in obtaining partial information from analyses even if a grammar is inadequate, or an expression ill-formed. A proof net matrix, though it may fail to include a proof net, includes all the modules, i.e. partial analyses, that the grammar is able to construct and is, in this initial sense, robust. And in the area of *underspecification* one is interested in reasoning without resolution of all ambiguity. Again, a proof net matrix represents different readings without enumerating them, and may thus provide a level of representation suited to reasoning with uncertainty.

References

Aarts, E. and K. Trautwein: 1995, 'Non-associative Lambek categorial grammar in polynomial time', *Mathematical Logic Quarterly* **41**, 476–484.

Aho, A. and J. Ullman: 1972, *The Theory of Parsing, Translation, and Compiling*, Volume 1 *Parsing*, Prentice-Hall New Jersey.

van Benthem, J.: 1991, *Language in Action: Categories, Lambdas and Dynamic Logic*, Studies in Logic and the Foundations of Mathematics Volume 130, North-Holland, Amsterdam.

Buszkowski, W.: 1986, 'Generative capacity of non-associative Lambek Calculus', *Bull. Acad. Pol. Sci. (Math.)*, 507–516.

Girard, J-Y.: 1987, 'Linear Logic', *Theoretical Computer Science* **50**, 1–102.

Hepple, M.: 1992, 'Chart parsing Lambek grammars: modal extensions and incrementality', *Proceedings of COLING-92*.

König, E.: 1994, 'A hypothetical reasoning algorithm for linguistic analysis', *Journal of Logic and Computation* **4**, 1–19.

Moortgat, M.: 1992, 'Labelled Deductive Systems for categorial theorem proving', OTS Working Paper OTS–WP–CL–92–003, Rijksuniversiteit Utrecht, also in *Proceedings of the Eighth Amsterdam Colloquium*, Institute for Language, Logic and Information, Universiteit van Amsterdam.

Moortgat, M.: 1996, 'Categorial type logics', in J. van Benthem and A. ter Meulen (eds.) *Handbook of Logic and Language*, Elsevier, to appear.

Morrill, G.: 1994, *Type Logical Grammar: Categorial Logic of Signs*, Kluwer Academic Publishers, Dordrecht.

Pentus, M.: 1993, 'Lambek grammars are context free', *Proceedings of the Eighth Annual IEEE Symposium on Logic in Computer Science*, Montreal.

Roorda, Dirk: 1991, *Resource Logics: proof-theoretical investigations*, Ph.D. dissertation, Universiteit van Amsterdam.