

# Proof Nets and Modality

Marco Hollenberg

March 16, 1992

## Abstract

In the Categorical Grammar literature a number of modalities have been proposed to deal with various linguistic phenomena. Examples are Morrill's modality for intensional domains ([10]) and structural modalities. We want an algorithmic proof theory for categorical logics with such modalities. A Gentzen-style sequent calculus is computationally inadequate: it suffers from spurious ambiguity. What is needed is an unambiguous representation of proofs. For this purpose we choose Moortgat's proof net approach ([7]) and extend it to deal with modalities, inspired by Wallen's work ([13]) on modal logic. The extension can be seen as the basis for all categorical modalities. It is not enough, however, to cover the properties of the structural modalities, which allow local relaxations of structural requirements. The paper will close with some suggestions on further extensions to such structural modalities and to multi-modal systems.

## 1 Linguistic motivation

In 1958 Lambek [5] introduced a calculus of syntactic types, which lies at the heart of Categorical Grammar today. The calculus captures the basic function-argument structure that seems present in natural language. It also captures the fact that many functors look for their arguments in a specific place, either to the left or to the right of the functor in question. So we have two type constructors:  $\backslash$  and  $/$ , capturing the notions of left-incompleteness and right-incompleteness respectively. An expression (a string) is of the functional type  $B \backslash A$  iff whenever it is concatenated with an expression of type  $B$  to the left it returns a type  $A$ . Likewise, an expression of type  $A/B$  is a functor that, given any argument of type  $B$  returns an expression of type  $A$ , but this time the argument must be concatenated to the right of the functor. Because the meaning of both type constructors is defined using the notion of concatenation, it seems reasonable to include one more type constructor:  $\bullet$ . A string is of type  $A \bullet B$  if the string is the concatenation of two other strings, of type  $A$  and  $B$  respectively.

Categorical Grammar is an optimal grammar formalism to implement Montague's program for compositional semantics, because the function-argument structure he saw in natural language semantics has a direct parallel in the syntactic types. For instance, the expression 'walks' is, syntactically, a functor  $np \backslash s$  that when concatenated to an argument of type  $np$  (a noun phrase) to its left returns an expression of type  $s$  (a sentence). Semantically, it is of type

$e \rightarrow t$ , a function from entities (type  $e$ ) to truth-values (type  $t$ ). When this function is applied to the semantics of the noun phrase (stipulated to be of type  $e$ ) it gives the meaning of the sentence (of truth-value type). The correspondence between the semantic type-system and the syntactic one is given by the following definition of the function  $TYP$ , that, given a syntactic type, returns the semantic type it corresponds to.

1. For  $a$  an atomic type,  $TYP(a)$  is stipulated. For instance:  $TYP(np) = e$  or  $(e \rightarrow t) \rightarrow t$ ,  $TYP(s) = t$ .
2.  $TYP(A/B) = TYP(B \setminus A) = TYP(B) \rightarrow TYP(A)$ .
3.  $TYP(A \bullet B) = TYP(A) \times TYP(B)$ , that is: the cartesian product.

Van Benthem [1] and Moortgat [6] extend Lambek's original calculus to calculate semantic recipes, using this strict parallel between syntax and semantics (see figure 1). This calculus will be referred to as **L**. The lexicon now consists of a finite list of assignments to words of natural language (i.e. syntactic atoms), of the form  $t : A$ , where  $A$  is a syntactic type and  $t$  is a semantic recipe of semantic type  $TYP(A)$ . These are the basic objects of the calculus. The calculus is in sequent-format: it operates on sequents of the form  $\Gamma \Rightarrow t : A$ , with  $\Gamma$  a non-empty sequence of semantics:syntax pairs, which means "from  $\Gamma$ ,  $t : A$  is deducible".  $\Gamma$  we call the *antecedent*,  $t : A$  the *succedent*. For each type constructor there are two rules: one for occurrences in the antecedent (rules of use, or left-rules, as indicated by the  $L$  in their names) another for occurrences in the succedent (rules of proof, or right-rules).

The calculus can be used for parsing as follows: Given a sequence of words  $\alpha_1 \dots \alpha_n$ , to deduce whether this is an expression of type  $B$  (for parsing purposes this will mostly be the sentence type  $s$ ), each word  $\alpha_i$  is linked via the lexicon to a labelled formula  $t_i : A_i$  and subsequently it is checked by the calculus whether  $t_1 : A_1, \dots, t_n : A_n \Rightarrow T : B$ <sup>1</sup> is a valid sequent of the calculus. This is done by using the rules in a backward fashion, from conclusion to premises. If it is valid, the semantic recipe for the whole expression,  $T$ , is also calculated.

In the figure, *terms* of the semantic object language are indicated by lowercase letters, meta-variables, which can be unified with *some* semantic recipe, by uppercase. Such meta-variables indicate the unknown parts of semantic recipes. Because the intended use of the algorithm is parsing, antecedent terms will be given, whereas succedent ones will not. Therefore, unification of semantic recipes in the axiom case is directional: the succedent one, which is a meta-variable, will be instantiated ( $:=$ ) to the semantic recipe in the antecedent.

The right-rules are slightly more complicated than presented. For instance, in the  $/R$ -rule, what really happens is: in the conclusion sequent, the succedent semantic recipe will be a meta-variable  $U$ . A new meta-variable,  $T$ , is chosen, and  $U := \lambda x.T$ . Proof search continues with the  $T$ .

$\pi_0$  and  $\pi_1$  are projection on the first element and the second element of pairs, respectively.

<sup>1</sup>The convention of using uppercase for unknowns and lowercase for knowns will be adhered to throughout this paper. One exception, however, will be the syntactic types, where a lowercase letter indicates an atomic type and uppercase an arbitrary type.

$$\begin{array}{c}
\text{Axiom:} \quad (T : A) \Rightarrow (U : A) \quad \{U := T\} \\
\\
\frac{\Delta \Rightarrow (T : A) \quad \Gamma, (T : A), \Theta \Rightarrow (U : C)}{\Gamma, \Delta, \Theta \Rightarrow (U : C)} \text{Cut} \\
\\
\frac{\Delta \Rightarrow (X : B) \quad \Gamma, (tX : A), \Theta \Rightarrow (U : C)}{\Gamma, (t : A/B), \Delta, \Theta \Rightarrow (U : C)} /L \quad \frac{\Gamma, (x : B) \Rightarrow (T : A)}{\Gamma \Rightarrow (\lambda x. T : A/B)} /R \\
\\
\frac{\Delta \Rightarrow (X : B) \quad \Gamma, (tX : A), \Theta \Rightarrow (U : C)}{\Gamma, \Delta, (t : B \backslash A), \Theta \Rightarrow (U : C)} \backslash L \quad \frac{(x : B), \Gamma \Rightarrow (T : A)}{\Gamma \Rightarrow (\lambda x. T : B \backslash A)} \backslash R \\
\\
\frac{\Gamma, (\pi_0 t : A), (\pi_1 t : B), \Theta \Rightarrow (U : C)}{\Gamma, (t : A \bullet B), \Theta \Rightarrow (U : C)} \bullet L \quad \frac{\Gamma \Rightarrow (T : A) \quad \Theta \Rightarrow (U : B)}{\Gamma, \Theta \Rightarrow ((T, U) : A \bullet B)} \bullet R
\end{array}$$

Figure 1: The Lambek calculus.

We can regard the calculus as defining a logic. It is a *substructural* logic because the antecedent is not a set of formulae, but a list and as such it is both order-conscious and resource-conscious, as indeed natural language itself is. So ‘John walks’ may be grammatical, even though neither ‘walks John’ (order-sensitivity) nor ‘John walks walks’ are (resource-sensitivity:  $np, np \backslash s$  is not the same as  $np, np \backslash s, np \backslash s$ , as it would be in classical logic, where stating the same formula many times makes no difference).  $/$  and  $\backslash$  can now be viewed as directional forms of implication. We find that the objects of the logic are *labelled* formulae, in the sense of Gabbay [2], with the semantics as the label of the syntactic formula. The calculus operates not just on the syntactic formulae but also on the semantic labels. As a logic, we want the derivability-arrow  $\Rightarrow$  to be both reflexive and transitive. For that purpose the Axiom and the Cut-rule, respectively, are included in the calculus.

If distinct object variables are used as semantic labels at the start of proof search, succedent semantic labels returned by the calculus can be seen to encode essential parts of proofs. Therefore we will sometimes refer to the semantic labels as ‘proof terms’. If two different proofs of a sequent give the same proof term (or equivalent proof terms), we say they only differ in inessential ways. Proofs that give two inequivalent proof terms differ essentially. More will be said about this later.

Morrill [10] adds as a new type constructor the modality  $\Box$  to capture intensional aspects of natural language semantics, which in categorial grammar should be reflected in the syntactic type assignments, because of the parallel architecture of syntax and semantics. To the definition of *TYP* we add that  $TYP(\Box A) = s \rightarrow TYP(A)$ , a function from type  $s$  objects (possible worlds, states of affairs) to  $TYP(A)$  objects. That we can construct functions such as these does not mean that type  $s$ -terms figure in the semantic language, unless stipulated as the semantic type of some basic (atomic) syntactic type. The possibility of constructing types  $\Box A$  corresponds in Montagovian semantics to

the possibility of constructing  $\tilde{t}$  and  $\tilde{t}$  terms, using the operation of intensionalization (abstracting over possible worlds) and extensionalization (applying a term to the present world), respectively. As a side effect the modality seems able to capture domain-sensitivity in syntax.

The sequent rules are categorial versions of those of S4 in classical modal logic. The classical versions allow discarding of material. Categorial versions, because of the resource-consciousness of language (every part of the linguistic expression plays a role in the final analysis), should not allow this.  $\Box\Gamma$  denotes a sequence  $t_1 : \Box A_1, \dots, t_n : \Box A_n$ , a fully modalized sequence.

$$\frac{\Gamma, \tilde{t} : A, \Theta \Rightarrow U : B}{\Gamma, t : \Box A, \Theta \Rightarrow U : B} \Box L \quad \frac{\Box\Gamma \Rightarrow T : B}{\Box\Gamma \Rightarrow \tilde{T} : \Box B} \Box R$$

With this modality we have the means to rule out (2) as ungrammatical, but still judge (1) to be grammatical.

- (1) Mary thinks John loves himself.
- (2) \* John thinks Mary loves himself.

Just using the language of **L**, plausible type-assignments (disregarding semantics) would be:

Mary	–	$np^f$
John	–	$np^m$
loves	–	$(np^X \backslash s) / np^Y$
thinks	–	$(np^X \backslash s) / s$
himself	–	$((np^m \backslash s) / np^m) \backslash (np^m \backslash s)$

Here a type  $np^g$  is a syntactic type with the gender feature set to  $g$ .  $f$  signifies feminine gender,  $m$  masculine and  $X$  and  $Y$  signify gender values that are not as of yet specified. Now, both (1) and (2) become derivable. This is because in (2), the types associated with ‘thinks Mary loves’ can be used to prove  $(np^m \backslash s) / np^m$ . Therefore, these can serve as the argument of ‘himself’. The modality allows us to assign lexical types in such a way that ‘himself’ can only look for its argument within its own clausal domain, thus introducing domain-sensitivity in Categorial Grammar.

Mary	–	$\Box np^f$
John	–	$\Box np^m$
loves	–	$\Box((np^X \backslash s) / np^Y)$
thinks	–	$\Box((np^X \backslash s) / \Box s)$
himself	–	$\Box(((np^m \backslash s) / np^m) \backslash (np^m \backslash s))$

The crucial change is found in the type-assignment for ‘thinks’. This now demands an  $s$ -type domain to its right. We find (1) to be derivable as an  $s$ , but not (2), with the present type-assignments. Trying the same trick as before will not help. At some stage of proof search we would encounter the query whether  $\Box np^f, \Box((np^X \backslash s) / np^Y), np^m \Rightarrow \Box s$ . This is not a valid sequent,

precisely because of the constraint on the right-rule for  $\Box$  that the antecedent must be fully modalized.

Besides this intensional modality, *structural* modalities have been suggested in the Categorical Grammar literature ([11, 9]). The idea is imported from Linear Logic. Structural rules are non-logical rules that destroy structure. Since in Categorical Grammar only the antecedent is assumed to be structured, this means: they destroy the structure of the antecedent. If we want to exploit the idea that some linguistic items relax certain structural constraints, whereas in general these constraints should not be violated, simple addition of structural rules will not do, because these simply discard of the structural constraint in question. The answer is to allow *type-controlled* structural rules, structural rules that can be controlled *lexically*. For this purpose structural modalities  $\Box_s$  can be used. Structural modalities have the rules of S4 as their basis, just like the intensional modality. In the case of structural modalities these rules have no semantic effect. It should therefore not come as a surprise that  $TYP(\Box_s A) = TYP(A)$ , for any structural modality  $\Box_s$ . In addition, there is a *licensed* structural rule for each structural modality.

Let us give an example. Order is a structural dimension that is clearly of importance to natural language. This is why **L** is order-sensitive. Sometimes, however, this order-sensitivity needs to be relaxed. Complete order-insensitivity could be achieved by adding the structural rule of permutation  $P$ , giving us the Lambek-van Benthem calculus **LP**:

$$\frac{\Gamma, (u : B), (t : A), \Theta \Rightarrow (T : C)}{\Gamma, (t : A), (u : B), \Theta \Rightarrow (T : C)} P$$

Such a rule is called a structural rule because all it does is make a statement about the structure of the antecedent. The same effect could be achieved by switching to a calculus where antecedents are multisets, instead of lists. Of-course, what would be more interesting is some restricted form of permutation, because complete order-insensitivity is not very realistic, linguistically speaking. For this purpose Moortgat and Morrill [9] use the structural modality of permutation  $\Box_p$ . This modality has the rules of S4 as before (but without the semantic operations) and a licensed rule of permutation, which has the same form as the unrestricted rule of permutation above, but with the restriction that either  $A$  or  $B$  is of the form  $\Box_p D$ . So, the modality licenses permutation.

One linguistic phenomenon that the permutation modality is useful for is non-peripheral extraction. The relative pronoun 'who' needs a sentential argument that is missing a noun phrase *somewhere* within it to form a noun modifier. **L** cannot express this: it can at most represent expressions that are missing something on their left, or on their right, never *within* it. The richer language with  $\Box_p$  gives us the required expressive power. The type assignment we're looking for is:  $(n \setminus n) / (s / \Box_p np)$ . We now find 'the man who Mary loves -' (right-peripheral extraction), 'the man who - loves Mary' (left-peripheral extraction) and 'the man who Mary loved - during the summer' (non-peripheral extraction) all derivable noun phrases with the single type assignment for 'that'.

We give part of the derivation for the final sentence, the case of non-peripheral extraction, to close off the section. ‘During the summer’ is assumed to be a verb phrase modifier  $(np \setminus s) \setminus (np \setminus s)$ . The crucial step in this derivation is the  $\Box_P$  step, which makes sure ‘loved’ gets its argument in the right position.

$$\begin{array}{c}
\frac{np \Rightarrow np}{\Box_P np \Rightarrow np} \Box_P L \quad \frac{\vdots}{np, np \setminus s, (np \setminus s) \setminus (np \setminus s) \Rightarrow s} / L \\
\frac{np, (np \setminus s) / np, \Box_P np, (np \setminus s) \setminus (np \setminus s) \Rightarrow s}{np, (np \setminus s) / np, (np \setminus s) \setminus (np \setminus s), \Box_P np \Rightarrow s} \Box_P P \\
\frac{np, (np \setminus s) / np, (np \setminus s) \setminus (np \setminus s) \Rightarrow s / \Box_P np}{np / n, n, (n \setminus n) / (s / \Box_P np), np, (np \setminus s) / np, (np \setminus s) \setminus (np \setminus s) \Rightarrow np} / R \quad \frac{\vdots}{np / n, n, n \setminus n \Rightarrow np} / L
\end{array}$$

If we’re working with binary trees as antecedents, a linguistically plausible strategy, an interesting structural modality would be one of associativity, which could license a local switch from binary trees to lists (see Moortgat [8]).

The remaining sections will be concerned solely with the computational issue of proof search in the calculus extended with S4-modalities. The next section introduces the basic problems, which will be attacked in the sections thereafter.

## 2 Computational issues

The Gentzen-style sequent formulation given in the previous section raises several questions about the usability of the calculus in parsing. Is validity of sequents a decidable problem? If so, is the sequent calculus efficient as a proof search algorithm? Is the inclusion of Cut crucial to the system?

This latter question is easily answered: no.

**Theorem:** Cut is eliminable in  $\mathbf{L} + \{\Box, \Box_P\}$ .

**Proof:** We will merely describe the proof, as the proof itself is not unlike other Cut-elimination proofs and therefore straightforward. Each Cut-rule occurrence

$$\frac{\frac{1}{\Delta \Rightarrow (t : A)} \quad \frac{2}{\Gamma, (t : A), \Delta \Rightarrow (u : C)}}{\Gamma, \Delta, \Theta \Rightarrow (u : C)} \text{Cut}$$

is given a certain metric degree, namely:  $\text{degree}(\text{Cut}) = d(\Gamma, \Delta, \Theta, A, C)$  where  $d$  counts the number of connectives in a sequence of formulae or types. We then construct a proof-reduction algorithm that takes any subproof that is an instance of Cut, but where its premisses **1** and **2** are Cut-free proofs, and return an equivalent subproof where

1. the Cut is replaced by Cut(s) of lower degree,
2. the Cut is of the same degree but is moved up, closer to the leaves of the proof tree, or
3. the Cut is removed.

This second case is mentioned because of reductions involving  $\Box_p P$  rules, such as the following:

$$\frac{\frac{1}{\Box_p \Delta \Rightarrow (t : A)} \quad \Box_p R \quad \frac{2}{\Gamma, (w : B), (t : \Box_p A), \Theta \Rightarrow (u : C)} \quad \Box_p P}{\Gamma, \Box_p \Delta, (w : B), \Theta \Rightarrow (u : C)} \text{Cut}$$

$\Downarrow$

$$\frac{\frac{1}{\Box_p \Delta \Rightarrow (t : A)} \quad \Box_p R \quad \frac{2}{\Gamma, (w : B), (t : \Box_p A), \Theta \Rightarrow (u : C)}}{\Gamma, (w : B), \Box_p \Delta, \Theta \Rightarrow (u : C)} \text{Cut} \quad n \times \Box_p P$$

Here the step annotated with  $n \times \Box_p P$  indicates  $n$  applications of  $\Box_p P$ ,  $n$  being the number of formulae in  $\Box_p \Delta$ . Every proof reduction step with this degree preserving behaviour has the property that the Cut is carried over a  $\Box_p P$  rule and therefore moved closer to the leafs. As proof trees are finite, no infinite chains of degree preserving steps exist. Therefore, the algorithm will at some stage replace the Cut by Cut(s) of lower degree, or it will eliminate the Cut completely. The latter will occur when one of the premises of the Cut is an instance of the axiom:

$$\frac{(t : A) \Rightarrow (t : A) \quad \frac{2}{\Gamma, (t : A), \Theta \Rightarrow (u : C)}}{\Gamma, (t : A), \Theta \Rightarrow (u : C)} \text{Cut}$$

$\Downarrow$

$$\frac{2}{\Gamma, (t : A), \Theta \Rightarrow (u : C)}$$

As the algorithm covers all Cut instances and as it terminates, we know it will return a Cut-free proof for every proof. The existence of this algorithm proves that Cut is eliminable. The proof terms that are lost by this procedure are equivalent to proof terms that are not lost, under the equational theory including the  $\beta$ - and  $\eta$ -elimination axioms and the projection axioms  $\pi_0(x, y) = x$  and  $\pi_1(x, y) = y$ .  $\square$

The Cut rule is the only rule in our calculus that has material in its premises that does not occur in its conclusion sequent. As Cut is eliminable, the calculus has the *subformula property*: in the proof of a sequent, nothing is needed besides the material in the sequent we're trying to prove.

Another result connected to the eliminativity of Cut is the decidability of the problem of validity. Every sequent rule, except  $\Box_p P$ , has less connectives in its premises than in its conclusion.  $\Box_p P$  permutes a type to another location in the

sequent. For each finite sequence there are only finitely many permutations, so no new sequents can be proved by allowing infinite chains of  $\Box_p P$ -applications. The sequent calculus may be used directly as a proof search algorithm, provided we add a control mechanism that makes sure the particular permutation proof search is dealing with at each moment has not occurred already, earlier in the proof.

That we *can* use the calculus given for proof search does not mean that we should. It is very inefficient and allows massive redundancies. First of all there is the copying of material after rule application. Consider  $/R$ . The antecedent of the conclusion sequent is completely copied to the premise, even though it is unchanged. Another, more serious redundancy is what has been termed *spurious ambiguity*: the *irrelevant* ordering of rule applications, an ordering being relevant only if this yields a different proof term than another ordering. A simple example is the sequent  $f : a/b, g : b/c \Rightarrow T : a/c$ , which has two sequent proofs, but only one proof term. Each proof contains two  $/L$  applications, once to  $a/b$ , once to  $b/c$ . The only difference between the proofs is the ordering of these applications. This is an irrelevant difference, however, since the same proof term is yielded,  $T = \lambda x.f(gx)$ .

Modalities give rise to spurious ambiguity of their own. Let us consider some examples:

- $\Box np, \Box(np \backslash s) \Rightarrow \Box s$ . Three proofs, one proof term. Irrelevance in the ordering of the  $\Box L, \backslash L$  rules can be seen here.

$$\begin{array}{c}
 \frac{np \Rightarrow np \quad s \Rightarrow s}{np, np \backslash s \Rightarrow s} \backslash L \\
 \frac{}{\Box np, np \backslash s \Rightarrow s} \Box L \\
 \frac{}{\Box np, \Box(np \backslash s) \Rightarrow s} \Box L \\
 \frac{}{\Box np, \Box(np \backslash s) \Rightarrow \Box s} \Box R
 \end{array}
 \qquad
 \begin{array}{c}
 \frac{np \Rightarrow np \quad s \Rightarrow s}{np, np \backslash s \Rightarrow s} \backslash L \\
 \frac{}{\Box np, np \backslash s \Rightarrow s} \Box L \\
 \frac{}{\Box np, \Box(np \backslash s) \Rightarrow s} \Box L \\
 \frac{}{\Box np, \Box(np \backslash s) \Rightarrow \Box s} \Box R
 \end{array}$$

$$\begin{array}{c}
 \frac{np \Rightarrow np}{\Box np \Rightarrow np} \Box L \\
 \frac{}{\Box np, np \backslash s \Rightarrow s} \backslash L \\
 \frac{}{\Box np, \Box(np \backslash s) \Rightarrow s} \Box L \\
 \frac{}{\Box np, \Box(np \backslash s) \Rightarrow \Box s} \Box R
 \end{array}$$

- $(np \backslash s)/ap, \Box_p np, \Box_p ap \Rightarrow s$ . Without any control-mechanism there are infinitely many proofs: application of  $\Box_p P$  gives us a sequent containing, once again a  $\Box_p$ -type, making another application of the rule possible, ad infinitum. A control-mechanism that does not try the same sequent twice makes the number of proofs finite, but still quite large, considering there is only one proof term to be found. We will give just two proofs, one quite short, the other stretched out some more. The reader can finish this second proof and verify that the control-mechanism mentioned will not rule it out as a proof: each sequent is considered at most once.



$$\begin{array}{c}
\frac{np \Rightarrow np \quad s \Rightarrow s}{ap \Rightarrow ap \quad np, np \setminus s \Rightarrow s} \setminus L \\
\frac{\quad}{np, (np \setminus s) / ap, ap \Rightarrow s} / L \\
\frac{\quad}{np, (np \setminus s) / ap, \Box_p ap \Rightarrow s} \Box_p L \\
\frac{\quad}{\Box_p np, (np \setminus s) / ap, \Box_p ap \Rightarrow s} \Box_p L \\
\frac{\quad}{(np \setminus s) / ap, \Box_p np, \Box_p ap \Rightarrow s} \Box_p P \\
\vdots \\
\frac{\quad}{\Box_p np, (np \setminus s) / ap, ap \Rightarrow s} \Box_p L \\
\frac{\quad}{\Box_p np, (np \setminus s) / ap, \Box_p ap \Rightarrow s} \Box_p P \\
\frac{\quad}{\Box_p np, \Box_p ap, (np \setminus s) / ap \Rightarrow s} \Box_p P \\
\frac{\quad}{\Box_p ap, \Box_p np, (np \setminus s) / ap \Rightarrow s} \Box_p P \\
\frac{\quad}{\Box_p ap, (np \setminus s) / ap, \Box_p np \Rightarrow s} \Box_p P \\
\frac{\quad}{(np \setminus s) / ap, \Box_p ap, \Box_p np \Rightarrow s} \Box_p P \\
\frac{\quad}{(np \setminus s) / ap, \Box_p np, \Box_p ap \Rightarrow s} \Box_p P
\end{array}$$

One approach to spurious ambiguity has been the *proof normalization* approach, as undertaken by Hepple [4] and Hendriks and Roorda [3]. We can define equivalence classes of proofs, such that proofs are in the same equivalence class only if they have the same antecedent and the same succedent type at their roots and if they produce equivalent proof terms (for instance: equivalence as defined by  $\alpha$ -,  $\beta$ -,  $\eta$ - and  $\hat{\cdot}$ -reduction, the projection axioms and by SP (surjectivity of pairing):  $(\pi_1 t, \pi_2 t) = t$ ). The proof normalization approach defines a normalform over proofs such that only one proof in each equivalence class is in this normalform. Ultimately it wants to devise restricted proof search methods that only return such normalform proofs. For  $L\{-\bullet\}$  this has been done. For instance, Hepple [4] introduces a proof search algorithm for  $L\{-\bullet\}$  that deterministically applies all  $/R, \setminus R$  rules until it has an atomic type in the succedent. This has as a side effect that no axiom sequents with non-atomic (complex) types will occur in any proof. Then, a type is sought in the antecedent that is to be unfolded completely by means of the left rules. There is no loss of valid sequents, nor of proof terms, as is proved in his dissertation. The copying redundancy is not dealt with of course: this is an inherent feature of any sequent approach.

For  $L+\{\Box\}$  no such approach has been worked out, to my knowledge. Let us examine a naive extension of Hepple's approach.

Such an extension would predict that the sequent  $t : \Box a \Rightarrow T : \Box a$  ( $a$  atomic) now gets only one proof term, namely  $T = \hat{t}$ .  $T = t$  is not returned. This is because Hepple's approach restricts the axiom to the atomic case, which the given sequent is not an instance of. This result is only wanted if we can somehow restrict semantic labels  $t$  occurring in the antecedent to be intensionally closed expressions, because then  $t = \hat{t}$  is valid and no real loss of proof terms occurs. Let us assume this to be the case, for convenience.

Consider the sequent  $\Box(a/b), (\Box b)/c, c \Rightarrow \Box a$ . The approach under consideration would first of all, force us to apply  $\Box R$ , to get an atomic antecedent.

This is not possible, because the type  $c$  bears no modality. Fair enough, let us then find an antecedent type to unfold completely. No choice would result in a valid sequent, however.

Since the proof normalization approach is not straightforwardly extendable, let us try our hand at another approach to spurious ambiguity: the proof net approach. The next section will introduce the notion of a proof net as it has been proposed for  $L$ , the one following that will consider what can be done to extend it to a syntactic language with modalities.

### 3 Proof nets

Another approach to the problem of spurious ambiguity is to leave the sequent format altogether and look for a more efficient proof representation, inherently redundancy-free. Such a representation is the proof net, as adapted by Roorda [12] from Linear Logic, and recast by Moortgat [7] in a Labelled Deductive Systems format. We give here, and extend, Moortgat's approach.

The proof net approach decomposes types into their atomic subtypes by means of *logical links* and then establishes *axiom links* between these. The type-decomposition corresponds to the use of left- or right- rules in the sequent proof search. Each axiom linking must correspond to axiom sequents in a proof of the corresponding sequent. To ensure this, we must label each type in the proof object with information about its location: whether it is an antecedent-type or a succedent-type. Antecedent-types are labelled with polarity 1 (types to be *used*), succedent types with polarity 0 (types to be *proved*). Logical links propagate these polarities as should be expected from the sequent rules. Consider the following logical link:

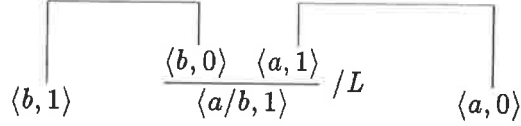
$$\frac{\langle A, 1 \rangle \quad \langle B, 0 \rangle}{\langle A/B, 1 \rangle} /L \quad \frac{\Delta \Rightarrow B \quad \Gamma, A, \Theta \Rightarrow C}{\Gamma, A/B, \Delta, \Theta \Rightarrow C} /L$$

$\langle A/B, 1 \rangle$  denotes an antecedent occurrence of a type  $A/B$ . Decomposition of such a type corresponds to the  $/L$ -rule in the sequent calculus. This tells us that decomposition gives us a succedent-type  $B$  ( $\langle B, 0 \rangle$ ) and an antecedent-type  $A$  ( $\langle A, 1 \rangle$ ). Axiom links connect an atomic antecedent-type with a succedent occurrence of the same type:

$$\begin{array}{c} \text{---} \\ | \quad | \\ \langle a, 1 \rangle \quad \langle a, 0 \rangle \end{array} \quad a \Rightarrow a$$

Proof search using this proof representation comes down to labelling the types in the sequent appropriately (up to now this means labelling each type with its correct polarity) unfolding the types into their atomic subtypes and finally finding an axiom linking such that every atomic subtype is linked to exactly one other. Every such decomposition and axiom linking results in a *proof structure*. Not every proof structure corresponds to a valid sequent however. We want to find a subset of the set of proof structures, the set of *proof nets* such that these correspond exactly to the valid sequents. To give an example of a proof

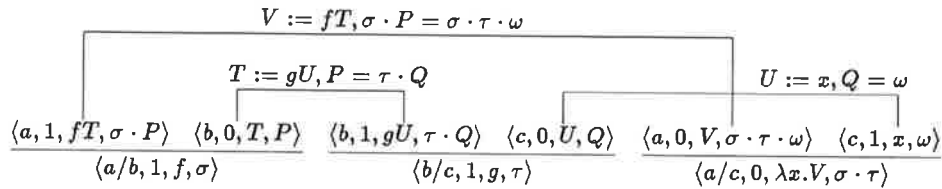
structure that is not a proof net, consider the following for the  $\mathbf{L}$ -invalid sequent  $b, a/b \Rightarrow a$ <sup>2</sup>:



For reasons such as this, Moortgat uses structure labelling in his proof nets. These labels are meant to reflect the structure of antecedents. In the present case, we can model the structure using strings, as antecedents are presumed to be lists in  $\mathbf{L}$ , which have the same properties as strings. Strings can be constructed from PAR and VAR (sets of string parameters and string variables respectively), by means of the associative concatenation operator  $\cdot$ , reflecting the associative  $\bullet$ -type constructor. One can now add structure labels to types in such a way that the concatenation of the structure labels in the antecedent is the same as the structure label of the succedent type, in a valid sequent. Figure 2 gives the links for  $\mathbf{L}$ , together with the corresponding rules in the sequent calculus. The sequent rules are augmented with structure labels, to explain the structure labels in the proof net links. The proof links link formulae of the form:  $\langle \text{Type}, \text{Polarity}, \text{Semantics}, \text{Structure} \rangle$ . The semantics label is dealt with in an entirely analogous way to the sequent calculus. Structure variables cannot unify with the empty string  $\varepsilon$ , which corresponds to the choice in  $\mathbf{L}$  to disallow empty antecedents.  $P, Q, R, S$  will be used as string variables,  $\sigma, \tau, \omega$  as string parameters, and  $\varphi, \psi, \chi$  as arbitrary strings.

To decide on the validity of a sequent  $x_1 : A_1, \dots, x_n : A_n \Rightarrow T : B$  we must build a proof net out of  $\{\langle A_1, 1, x_1, \sigma_1 \rangle, \dots, \langle A_n, 1, x_n, \sigma_n \rangle, \langle B, 0, T, \sigma_1 \dots \sigma_n \rangle\}$  with  $\sigma_i$  distinct string parameters. The multiset before decomposition we call the *root path*, the one after decomposition the *atomic path*.

Let us give an example.  $a/b, b/c \Rightarrow a/c$  has two sequent proofs and one proof term. There is just one proof net:



Unifier:  $\{P \leftarrow \tau \cdot \omega, Q \leftarrow \omega, V \leftarrow f(gx), T \leftarrow gx, U \leftarrow x\}$ .

Proof term:  $\lambda x. f(gx)$ .

The proof structure we saw before now fails to be a proof net, because of a unification error (we can disregard semantics here, because adding this will not make the proof structure into a proof net):

<sup>2</sup>Just for this example, we let the ordering of the types in the sequent correspond to the ordering of the ordering of the formulae at the bottom of the proof structure. At no time will the ordering of premises be relevant. This is in sharp contrast to Roorda's approach, which makes crucial use of just this ordering.

Axiom link:

$$\frac{U := T, \varphi = \psi}{\langle a, 1, T, \varphi \rangle \quad \langle a, 0, U, \psi \rangle}$$

Axiom sequent:

$$a^\varphi \Rightarrow a^\psi$$

Logical links:

$$\frac{\langle A, 1, tT, \varphi \cdot P \rangle \quad \langle B, 0, T, P \rangle}{\langle A/B, 1, t, \varphi \rangle} /L$$

$$\frac{\langle A, 1, tT, P \cdot \varphi \rangle \quad \langle B, 0, T, P \rangle}{\langle B \setminus A, 1, t, \varphi \rangle} \setminus L$$

$$\frac{\langle A, 0, T, \varphi \cdot \sigma \rangle \quad \langle B, 1, x, \sigma \rangle}{\langle A/B, 0, \lambda x.T, \varphi \rangle} /R$$

$$\frac{\langle A, 0, T, \sigma \cdot \varphi \rangle \quad \langle B, 1, x, \sigma \rangle}{\langle B \setminus A, 0, \lambda x.T, \varphi \rangle} \setminus R$$

$$\frac{\langle A, 1, \pi_0 t, \varphi \rangle \quad \langle B, 1, \pi_1 t, \psi \rangle}{\langle A \bullet B, 1, t, \varphi \cdot \psi \rangle} \bullet L$$

$$\frac{\langle A, 0, T, \varphi \rangle \quad \langle B, 0, U, \psi \rangle}{\langle A \bullet B, 0, (T, U), \varphi \cdot \psi \rangle} \bullet R$$

Logical sequent rules:

$$\frac{\Delta^P \Rightarrow B^P \quad \Gamma^\psi, A^{\varphi \cdot P}, \Theta^x \Rightarrow C^{\psi \cdot \varphi \cdot P \cdot x}}{\Gamma^\psi, A/B^\varphi, \Delta^P, \Theta^x \Rightarrow C^{\psi \cdot \varphi \cdot P \cdot x}} /L$$

$$\frac{\Delta^P \Rightarrow B^P \quad \Gamma^\psi, A^{P \cdot \varphi}, \Theta^x \Rightarrow C^{\psi \cdot P \cdot \varphi \cdot x}}{\Gamma^\psi, \Delta^P, B \setminus A^\varphi, \Theta^x \Rightarrow C^{\psi \cdot P \cdot \varphi \cdot x}} \setminus L$$

$$\frac{\Gamma^\varphi, B^\sigma \Rightarrow A^{\varphi \cdot \sigma}}{\Gamma^\varphi \Rightarrow A/B^\varphi} /R$$

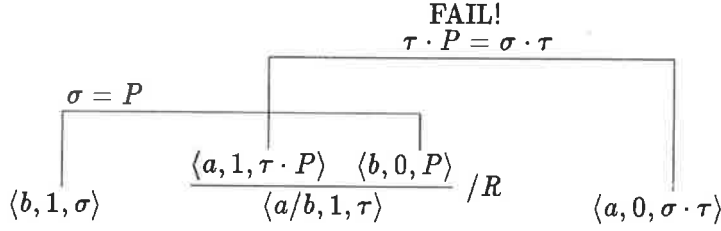
$$\frac{B^\sigma, \Gamma^\varphi \Rightarrow A^{\sigma \cdot \varphi}}{\Gamma^\varphi \Rightarrow B \setminus A^\varphi} \setminus R$$

$$\frac{\Gamma^x, A^\varphi, B^\psi, \Theta^\omega \Rightarrow C^{x \cdot \varphi \cdot \psi \cdot \omega}}{\Gamma^x, A \bullet B^{\varphi \cdot \psi}, \Theta^\omega \Rightarrow C^{x \cdot \varphi \cdot \psi \cdot \omega}} \bullet L$$

$$\frac{\Gamma^\varphi \Rightarrow A^\varphi \quad \Delta^\psi \Rightarrow B^\psi}{\Gamma^\varphi, \Delta^\psi \Rightarrow A \bullet B^{\varphi \cdot \psi}} \bullet R$$

$P$  a fresh string-variable,  $\sigma$  a fresh string-parameter.

Figure 2: Left: proof links for **L**. Right: corresponding sequent rules with structure (string) labels



No semantic readings are lost using the proof net approach. Different proof terms are gotten via different axiom linking. However, like Hepple's proof normalization, the proof net approach gives just one proof term for a complex axiom sequent such as  $x : a/b \Rightarrow T : a/b$ , namely  $T = \lambda y.xy$ .  $T = x$  is not found. This is because types are decomposed into their atomic subtypes: axiom links between complex types are ruled out. It is no real loss, however, since the two terms are  $\eta$ -equivalent.

The given proof links are essentially right, as can be seen from the corresponding annotated sequent rules. But from a proof search point of view, something is missing. The problem lies with the  $\bullet L$ -,  $\bullet R$ -links. These require the splitting in halves of the conclusion string label. Up to now, nothing has enforced this to be always possible. We will change some of the links to enforce precisely this. The altered links make use of a function  $f : TYPE \rightarrow N_{>0}$  ( $TYPE$  is the set of syntactic types constructed from  $/$ ,  $\backslash$  and  $\bullet$ ). This function is defined as follows (for  $a$  atomic):

$$\begin{aligned} f(a) &= 1 \\ f(A/B) &= f(B \backslash A) = f(A) \\ f(A \bullet B) &= f(A) + f(B) \end{aligned}$$

To determine the validity of a sequent  $x_1 : A_1, \dots, x_n : A_n \Rightarrow T : B$  we must build a proof net out of:

$$\begin{aligned} \{ & \langle A_1, 1, x_1, \sigma_{1,1} \dots \sigma_{1,f(A_1)} \rangle, \\ & \dots, \\ & \langle A_n, 1, x_n, \sigma_{n,1} \dots \sigma_{n,f(A_n)} \rangle, \\ & \langle B, 0, T, \sigma_{1,1} \dots \sigma_{1,f(A_1)} \dots \sigma_{n,1} \dots \sigma_{n,f(A_n)} \rangle \} \end{aligned}$$

with  $\sigma_{i,j}$  distinct string parameters. The links will make sure that for every tuple  $\langle A, 1, t, \varphi \rangle$  in the proof object  $\varphi = u\sigma_1 \dots \sigma_{f(A)}v$ , with  $u, v \in VAR^*$  and  $\sigma_i \in PAR$ .

### Altered links

$$\begin{aligned} & \frac{\langle A, 0, T, \varphi\sigma_1 \dots \sigma_{f(B)} \rangle \quad \langle B, 1, x, \sigma_1 \dots \sigma_{f(B)} \rangle}{\langle A/B, 0, \lambda x.T, \varphi \rangle} /R \\ & \frac{\langle A, 0, T, \sigma_1 \dots \sigma_{f(B)}\varphi \rangle \quad \langle B, 1, x, \sigma_1 \dots \sigma_{f(B)} \rangle}{\langle B \backslash A, 0, \lambda x.T, \varphi \rangle} \backslash R \end{aligned}$$

$$\frac{\langle A, 1, \pi_1 t, u\sigma_1 \dots \sigma_{f(A)} \rangle \quad \langle B, 1, \pi_2 t, \sigma_{f(A)+1} \dots \sigma_{f(A)+f(B)} \rangle}{\langle A \bullet B, 1, t, u\sigma_1 \dots \sigma_{f(A)+f(B)} v \rangle} \bullet L$$

The fact that we can choose a unique splitting in the  $\bullet L$ -link reflects the fact that, given an antecedent-type  $A \bullet B$ , a backward  $\bullet L$ -rule application gives us a unique choice of premiss. A backward  $\bullet R$ -application does not, in general, determine its premises. Therefore, it should not be expected that there is a unique splitting in the  $\bullet R$ -link. So we make no such choice at the time the  $A \bullet B$ -type is unfolded: we delay it, but make sure it is made at the time of the establishing of the axiom linking, by adding a constraint to unification:  $P \cdot Q = \varphi$ :

$$\frac{\langle A, 0, T, P \rangle \quad \langle B, 0, U, Q \rangle}{\langle A \bullet B, 0, (T, U), \varphi \rangle} \bullet R \quad \{P \cdot Q = \varphi\}$$

The proof net approach enables us to make the lexicon more efficient for parsing purposes, by means of partial execution. Given a lexical entry  $(\alpha, A, t)$ , where  $\alpha$  is the word in question,  $A$  its syntactic type, and  $t$  its semantic recipe, we can construct a partially executed lexical entry  $(\alpha, \sigma_1 \dots \sigma_{f(A)}, \Gamma, C)$ . Here,  $\Gamma$  is the atomic path we get after unfolding  $\langle A, 1, t, \sigma_1 \dots \sigma_{f(A)} \rangle$  by means of the logical links.  $\sigma_1, \dots, \sigma_{f(A)}$  are fresh parameters (i.e. not occurring anywhere else in the lexicon). Their concatenation must be stored in the lexical entry, to know, when parsing some linguistic utterance containing the word  $\alpha$ , what string label to assign to the succedent type. Finally,  $C$  is a set of constraints, one for every  $\bullet R$ -link used in the unfolding. These are added to the set of equations to be solved by unification at the time of axiom linking.

We can get from the original entry to the partially executed one deterministically, and since the latter contains all the information we need for parsing purposes, we can discard of the original one. So, the logical decomposition of the lexicon may be done at compile time. All that remains to be done at run time is:

1. The unfolding of the formula corresponding to the succedent-type of the sequent the validity of which we're interested in. This may be a trivial matter: practical applications will mostly parse sentences (type  $s$ ) on which no decomposition is possible.
2. The construction of axiom links.

Notice that nothing has been said in this section about the modal part of the logical language under consideration. This will be the topic of the next section.

## 4 Proof nets and modalities

In this section, we extend the notion of a proof net to cover  $\Box$ -types. The problem we encounter when we try this does not lie with the logical links. Logical

links just unfold types into their subtypes. In the modal case this comes down to unfolding a type  $\Box A$  into its single subtype  $A$ . Rather, the problem lies with the axiom links. Remember that any axiom link should correspond to an axiom sequent in a corresponding sequent proof, in a one-to-one fashion. Nothing said so far, however, forbids the following proof structure to be a proof net, even though the corresponding sequent is invalid:

$$\frac{\frac{\langle a, 1, \sigma \rangle \quad \frac{\langle a, 0, \sigma \rangle}{\langle \Box a, 0, \sigma \rangle} \Box R}{\langle a, 1, \sigma \rangle} \quad \frac{FAIL!}{a \Rightarrow \Box a}$$

Wallen ([13]) addresses this problem for the case of classical modal logic. He makes use of the semantics of modal logic: formulae are not just true or false, they are true or false with respect to some *point* in a Kripke structure. He labels his formulae accordingly, very much in the spirit of Gabbay's Labelled Deductive Systems, again. Let us investigate a proposal of this kind.

Suppose we want to prove  $\Box A \Rightarrow \Box \Box A$  in S4. One way to do this is to suppose  $\Box A$  is *true* at some point  $w_0$  in some S4 Kripke model  $\langle W, R, I \rangle$  ( $W$  a set of points,  $R \subseteq W \times W$ ,  $R$  reflexive and transitive, and  $I$  an interpretation function as usual) and that  $\Box \Box A$  is *false* at that same point. We should now be able to reach a contradiction, as  $\Box A \Rightarrow \Box \Box A$  is valid in S4. As  $w_0 \not\models \Box \Box A$  ( $\models$  is the valuation function over  $I$ ), there must be points  $w_1, w_2 \in W$  such that  $w_0 R w_1 R w_2$  and  $w_1 \not\models \Box A$ ,  $w_2 \not\models A$ . By transitivity of  $R$ :  $w_0 R w_2$ , and because  $w_0 \models \Box A$ :  $w_2 \models A$  must be the case. So we have reached a contradiction. This proof could be represented as:

$$\text{proof: } \frac{\frac{\frac{w_2 : A \Rightarrow w_2 : A}{w_0 : \Box A \Rightarrow w_2 : A} \Box L}{w_0 : \Box A \Rightarrow w_1 : \Box A} \Box R}{w_0 : \Box A \Rightarrow w_0 : \Box \Box A} \Box R \quad \text{configuration: } \begin{array}{ccc} w_0 & \xrightarrow{\quad} & w_2 \\ & \searrow \quad \nearrow & \\ & w_1 & \end{array}$$

An antecedent formula  $w : A$  can be read here as  $A$  is true at  $w$ , while such a succedent formula can be read as  $A$  is false at  $w$ . So the proof is an SLD-resolution proof. Sequent rules for proofsearch could manipulate both a sequent and a configuration:

$$\frac{w : A \Rightarrow w : A}{\Gamma, w_j : A, \Theta \Rightarrow w_k : B} \Box L \quad \frac{\Gamma \Rightarrow w_j : A}{\Gamma \Rightarrow w_i : \Box A} \Box R$$

*Condition on  $\Box L$ :*  $w_i R w_j$  in the present configuration.

*Operation at  $\Box R$ :* extend the present configuration with a new point  $w_j$ , such that  $w_i R w_j$ . Take the reflexive, transitive closure of the result as the new configuration.

Proving the sequent  $A_1, \dots, A_n \Rightarrow B$  would now be done by proving the labelled sequent  $w_0 : A_1, \dots, w_0 : A_n \Rightarrow w_0 : B$ , where the starting configuration is  $\langle \{w_0\}, \{\langle w_0, w_0 \rangle\} \rangle$ .

The configurations built during proof search turn out to be posets, that is:  $R$  is reflexive, transitive and *antisymmetric*<sup>3</sup>. Wallen exploits this by representing points as strings and letting  $R$  be the prefix relation  $\preceq$ . The reference to ‘the present configuration’ in the  $\Box L$ -rule can be dealt with by using variables. The sequent-rules have now become:

$$\frac{\{p = q\}}{p : A \Rightarrow q : A} Ax$$

$$\frac{\Gamma, p \cdot v : A, \Theta \Rightarrow q : B}{\Gamma, p : \Box A, \Theta \Rightarrow q : B} \Box L \quad \frac{\Gamma \Rightarrow p \cdot w : A}{\Gamma \Rightarrow p : \Box A} \Box R$$

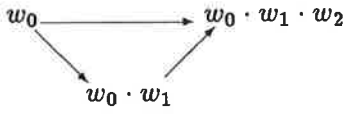
$w$  a fresh point parameter,  $v$  a fresh point variable.

The equation  $p = q$  in the axiom can be checked using some specialized string unification algorithm that allows variables to be unified with any string, empty or otherwise. The proof for  $\Box A \Rightarrow \Box \Box A$  now becomes:

$$\frac{\{w_0 \cdot v = w_0 \cdot w_1 \cdot w_2\}}{w_0 \cdot v : A \Rightarrow w_0 \cdot w_1 \cdot w_2 : A} Ax$$

$$\frac{w_0 \cdot v : A \Rightarrow w_0 \cdot w_1 \cdot w_2 : A}{w_0 : \Box A \Rightarrow w_0 \cdot w_1 \cdot w_2 : A} \Box L$$

$$\frac{w_0 : \Box A \Rightarrow w_0 \cdot w_1 \cdot w_2 : A}{w_0 : \Box A \Rightarrow w_0 \cdot w_1 : \Box A} \Box R$$

$$\frac{w_0 : \Box A \Rightarrow w_0 \cdot w_1 : \Box A}{w_0 : \Box A \Rightarrow w_0 : \Box \Box A} \Box R$$


Unifier:  $\{v \leftarrow w_1 \cdot w_2\}$ .

Notice that  $A \Rightarrow \Box A$  remains invalid in this system, as required:

$$\frac{\text{FAIL!}}{\{w_0 = w_0 \cdot w_1\}} Ax$$

$$\frac{w_0 : A \Rightarrow w_0 \cdot w_1 : A}{w_0 : A \Rightarrow w_0 : \Box A} \Box R$$

Notice also that if such point information is added to the proof structure at the beginning of this section, the desired unwellformedness is achieved. So we see already why this is a fruitful road to take.

The strings need not be looked at as points in Kripke frames. Point parameters can be associated with modalized succedent types. Point variables can be associated with modalized antecedent types. When a unifier tells us that a point variable must unify with a sequence of point parameters, this means, in the sequent calculus, that all the boxes associated with the point parameters

<sup>3</sup>Since the configurations constructed during proof search are posets, it seems we’ve descended into the logic of posets, which is known to be weaker than the logic of S4 (as was pointed out to me by Marcus Kracht). However, Wallen proves that his matrix system which is very similar to the above, is equivalent to the logic of S4, so the worry seems unfounded. See the remark above, on the notion of a reduction order.



in the sequence must be removed before the box associated with the variable is removed. So, unification of point values helps define a *reduction order*: it tells us which types must be unfolded before which others. This reduction order cannot be cyclic, as this would require us to apply a rule before we apply it! The acyclicity is related to the antisymmetric nature of the above configurations.

In the following sections we will explore how these insights can be applied to Categorical Grammar. Section 4.1 will show how easy it is to apply Wallen's approach to Morrill's pure S4-modality. Section 4.2 will discuss the implications for the structural modalities, while section 4.3 discusses applications to multi-modal systems.

#### 4.1 Morrill's modality

Morrill's modality, as we saw in section 1, is nothing but a categorical version of an S4-modality. Wallen's ideas should therefore be straightforwardly applicable to  $\mathbf{L} + \{\Box\}$ . Modal links should not alter the values of the structure component of formulae, as can be argued from the following presentation of the modal sequent rules:

$$\frac{\Gamma^\varphi, A^\psi, \Theta^x \Rightarrow B^{\varphi \cdot \psi \cdot x}}{\Gamma^\varphi, \Box A^\psi, \Theta^x \Rightarrow B^{\varphi \cdot \psi \cdot x}} \Box L \quad \frac{\Box \Gamma^\varphi \Rightarrow A^\varphi}{\Box \Gamma^\varphi \Rightarrow \Box A^\varphi} \Box R$$

The concatenation of the string values in the antecedent should be the same as the string values of the succedent. The  $\Box L$ -rule does not alter its succedent, nor its context. Therefore, both  $\Box A$  and  $A$  should cover the same string. The  $\Box R$ -rule maintains the antecedent as it is, so here the same story goes through.

Figure 3 gives the full presentation of the proposed system. The fundamental unit of proof search is now the labelled formula:  $\langle \text{Type, Polarity, Semantics, String, Point} \rangle$ . The first four places of this quintuple are as before. The point label is a string, as in the previous section.

Point labels are built using the non-commutative string concatenation operator  $\cdot$  and from two sets,  $\text{VARP}$  and  $\text{PARP}$ , containing point variables and point parameters respectively.  $v_i$  will be used for point variables,  $w_i$  for point parameters and  $p, q$  for arbitrary point labels. To prove the sequent  $x_1 : A_1, \dots, x_n : A_n \vdash T : B$  one has to build a proof net out of the multiset  $\{\langle A_1, 1, x_1, \sigma_1, w_0 \rangle, \dots, \langle A_n, 1, x_n, \sigma_n, w_0 \rangle, \langle B, 0, T, \sigma_1 \cdot \dots \cdot \sigma_n, w_0 \rangle\}$  with  $w_0 \in \text{PARP}$ . The  $\Box L$ -,  $\Box R$ -links act as in the sequent presentation in the previous section. Note that nonmodal links simply pass on the point value to their premisses.

Some examples will help to get acquainted with the system:

1.  $\Box np, \Box(np \backslash s) \Rightarrow \Box s$ . We disregard semantics here, for clarity's sake. Crucial is here that there is only one proof net, even though there are three sequent proofs for the sequent. Every one of these sequent proofs give the same proof term, so no reading is lost by the proof net approach.

$$\begin{array}{c}
\frac{}{U := T, \varphi = \psi, p = q} \\
\frac{}{\langle a, 1, T, \varphi, p \rangle \quad \langle a, 0, U, \psi, q \rangle} \\
\\
\frac{\langle A, 1, T, \varphi, p \cdot v \rangle}{\langle \Box A, 1, T, \varphi, p \rangle} \Box L \quad \{v \text{ a fresh element of VARP} \} \\
\\
\frac{\langle A, 0, T, \varphi, p \cdot w \rangle}{\langle \Box A, 0, T, \varphi, p \rangle} \Box R \quad \{w \text{ a fresh element of PARP} \} \\
\\
\frac{\frac{\langle A, 1, tT, \varphi \cdot P, p \rangle \quad \langle B, 0, T, P, p \rangle}{\langle A/B, 1, t, \varphi, p \rangle} /L \quad \frac{\langle A, 0, T, \varphi \cdot \sigma, p \rangle \quad \langle B, 1, x, \sigma, p \rangle}{\langle A/B, 0, \lambda x. T, \varphi, p \rangle} /R \\
\frac{\langle A, 1, tT, P \cdot \varphi, p \rangle \quad \langle B, 0, T, P, p \rangle}{\langle B \setminus A, 1, t, \varphi, p \rangle} \setminus L \quad \frac{\langle A, 0, T, \sigma \cdot \varphi, p \rangle \quad \langle B, 1, x, \sigma, p \rangle}{\langle B \setminus A, 0, \lambda x. T, \varphi, p \rangle} \setminus R \\
\\
\frac{\langle A, 1, \pi_0 t, \varphi, p \rangle \quad \langle B, 1, \pi_1 t, \psi, p \rangle}{\langle A \bullet B, 1, t, \varphi \cdot \psi, p \rangle} \bullet L \quad \frac{\langle A, 0, T, \varphi, p \rangle \quad \langle B, 0, U, \psi, p \rangle}{\langle A \bullet B, 0, (T, U), \varphi \cdot \psi, p \rangle} \bullet R
\end{array}$$

Figure 3: Proof links for  $\mathbf{L} + \{\Box\}$

$$\begin{array}{c}
\sigma = P, w_0 \cdot v_0 = w_0 \cdot v_1 \quad P \cdot \tau = \sigma \cdot \tau, w_0 \cdot v_1 = w_0 \cdot w_1 \\
\frac{\langle np, 1, \sigma, w_0 \cdot v_0 \rangle}{\langle \Box np, 1, \sigma, w_0 \rangle} \quad \frac{\langle np, 0, P, w_0 \cdot v_1 \rangle \quad \langle s, 1, P \cdot \tau, w_0 \cdot v_1 \rangle}{\langle np \setminus s, 1, \tau, w_0 \rangle} \quad \frac{\langle s, 0, \sigma \cdot \tau, w_0 \cdot w_1 \rangle}{\langle \Box s, 0, \sigma \cdot \tau, w_0 \rangle}
\end{array}$$

Unifier:  $\{P \leftarrow \sigma, v_0 \leftarrow w_1, v_1 \leftarrow w_1\}$

2.  $x : \Box a \Rightarrow T : \Box a$ . Notice that the only proof term found is  $\tilde{x}$ . This corresponds in the sequent calculus to a proof which unfolds the types to their atomic subtypes  $a$ . It is therefore crucial for our approach that the semantics uses intensionally closed expressions only.

$$\begin{array}{c}
\tilde{x} = U, \sigma = \sigma, w_0 \cdot v = w_0 \cdot w_1 \\
\frac{\langle a, 1, \tilde{x}, \sigma, w_0 \cdot v \rangle}{\langle \Box a, 1, x, \sigma, w_0 \rangle} \Box L \quad \frac{\langle a, 0, U, \sigma, w_0 \cdot w_1 \rangle}{\langle \Box a, 0, T, \sigma, w_0 \rangle} \Box R \{T = U\}
\end{array}$$

Unifier:  $\{T \leftarrow \tilde{x}, U \leftarrow \tilde{x}, v \leftarrow w_1\}$

3. Finally we give the proof structure the discussion started out with:

$$\begin{array}{c}
\text{FAIL!} \\
w_0 = w_0 \cdot w_1 \\
\frac{\langle a, 1, \sigma, w_0 \rangle \quad \frac{\langle a, 0, \sigma, w_0 \cdot w_1 \rangle}{\langle \Box a, 0, \sigma, w_0 \rangle} \Box R}{\langle \Box a, 1, \sigma, w_0 \rangle} \Box L
\end{array}$$

## 4.2 Structural Modalities

The system presented can serve as the basis for the structural modalities, since these have S4-rules in the sequent calculus. So, as long as we stay within an associative system, the  $\Box_i L$ - and  $\Box_i R$ -rules are taken care of. Nonassociative systems present problems of their own, which will have to be dealt with, as Moortgat [8] shows that such systems can be adequately motivated. This is, however, not the paper to do this. Let us look at a specific problem associated with structural modalities: they force us to redesign our proof structures, because they license relaxations of structural assumptions embodied in our original proof structures. The latter proof structures will consider *less* proofs as valid than wanted, judging from the sequent calculus. As the structure labelling of our formulae encodes these structural assumptions, we must alter this, without collapsing into a weaker system. Let us give an illustration of this.  $np \setminus s, \Box_p np \Rightarrow s$  is a valid sequent, as the following sequent proof tells us:

$$\frac{\frac{\frac{np \Rightarrow np}{\Box_p np \Rightarrow np} \Box_p L \quad s \Rightarrow s}{\Box_p np, np \setminus s \Rightarrow s} \setminus L}{np \setminus s, \Box_p np \Rightarrow s} \Box_p P$$

Still, a naive unfolding does *not* result in a proof net:

$$\begin{array}{c}
\text{FAIL!} \\
P \cdot \sigma = \sigma \cdot \tau \qquad w_0 = w_0 \cdot v \\
\frac{\frac{\langle s, 1, P \cdot \sigma, w_0 \rangle \quad \langle np, 0, P, w_0 \rangle}{\langle np \setminus s, 1, \sigma, w_0 \rangle} \setminus L \quad \frac{\langle np, 1, \tau, w_0 \cdot v \rangle}{\langle \Box_p np, 1, \tau, w_0 \rangle} \Box_p L}{\langle s, 0, \sigma \cdot \tau, w_0 \rangle} \Box L
\end{array}$$

even though the axiom linking corresponds to the axiom sequents in the sequent proof. The problem lies in that we chose string concatenation to be non-commutative. If we switch to a commutative one the proof structure becomes a proof net. However, letting string concatenation be commutative gives us **LP**, that is: **L** with the *unrestricted* rule of permutation. One option is to attach new labels to our formulae, that encode further order restrictions, which the present strings fail to encode.

Another option is to use complex string unification algorithms that can handle *permutable* strings. The  $\Box_p$ -links would then look as follows:

$$\frac{\langle A, 1, \Delta \varphi, p \cdot v \rangle}{\langle \Box_p A, 1, \varphi, p \rangle} \Box_p L \qquad \frac{\langle A, 1, \varphi, p \cdot w \rangle}{\langle \Box_p A, 1, \varphi, p \rangle} \Box_p R$$

where  $\triangle\varphi$  indicates a string  $\varphi$  which is permutable. The equation that fails in the proof structure above becomes  $\triangle\psi \cdot \varphi = \varphi \cdot \psi$ , which should now succeed because  $\psi$  can be moved past  $\varphi$  on the left of the equation. Notice that  $\Box_p R$  does not alter its structure label. This is because structural rules in categorial grammar only make statements about the structure of the antecedent, never about that of the succedent. This option of enriching string unification seems the one to choose, as it is easily generalizable to other structural modalities. One drawback is that the problem is displaced to the axiom-linking, where all the hard work is done. The kind of unification algorithms required and their properties will have to be explored.

### 4.3 Multi-modal systems

Full-fledged categorial grammars will need to use more than a single modality. One could have a variety of structural modalities, Morrill's modality and maybe others. Our point labels should have enough expressive power to differentiate between the various modalities. A first thing to notice is that, with two S4-modalities  $\Box_1$  and  $\Box_2$ ,  $\Box_1\Box_2 a \not\Rightarrow \Box_2\Box_1 a$ . A naive unfolding yields the following proof structure though:

$$\begin{array}{c}
 w_0 \cdot v_0 \cdot v_1 = w_0 \cdot w_1 \cdot w_2 \\
 \hline
 \frac{\langle a, 1, \sigma, w_0 \cdot v_0 \cdot v_1 \rangle}{\langle \Box_2 a, 1, \sigma, w_0 \cdot v_0 \rangle} \Box_2 L \quad \frac{\langle a, 0, \sigma, w_0 \cdot w_1 \cdot w_2 \rangle}{\langle \Box_1 a, 0, \sigma, w_0 \cdot w_1 \rangle} \Box_1 R \\
 \frac{\langle \Box_2 a, 1, \sigma, w_0 \cdot v_0 \rangle}{\langle \Box_1 \Box_2 a, 1, \sigma, w_0 \rangle} \Box_1 L \quad \frac{\langle \Box_1 a, 0, \sigma, w_0 \cdot w_1 \rangle}{\langle \Box_2 \Box_1 a, 0, \sigma, w_0 \rangle} \Box_2 R
 \end{array}$$

$w_0 v_0 v_1 = w_0 w_1 w_2$  succeeds under the unifier  $\{v_0 \leftarrow w_1 w_2, v_1 \leftarrow \varepsilon\}$  (for instance). To capture the fact that different modalities are involved we could *type* the variables and parameters in the point label. For each modality we should then have a different type associated with it (types are superscripted):

$$\frac{\langle A, 1, \varphi, p \cdot v^i \rangle}{\langle \Box_i A, 1, \varphi, p \rangle} \Box_i L \quad \frac{\langle \Box_i A, 0, \varphi, p \cdot w^i \rangle}{\langle \Box_i A, 0, \varphi, p \rangle} \Box_i R$$

Variables of one type we must require to be unifiable with parameters of that *same* type or with the empty string  $\varepsilon$ . The equation above now becomes  $w_0^0 \cdot v_0^1 \cdot v_1^2 = w_0^0 \cdot w_1^2 \cdot w_2^1$  and unification fails on this, correctly judging the above proof structure not to be a proof net.

Moortgat and Morrill [9] introduces modalities  $\Box_M$ , where  $M$  is a nonempty set of names for modalities. These *combined* modalities have the combined effect of all modalities  $\Box_s$ , with  $s \in M$ . Our previous modalities  $\Box_i$  would, in such a system, become  $\Box_{\{i\}}$ . In this system  $\Box_{M_1} a \Rightarrow \Box_{M_2} a$  iff  $M_2 \subseteq M_1$ . So, in the axiom linking of proof structures for this system:  $v^M = w_1^{M_1} \dots w_n^{M_n}$  iff  $M_1, \dots, M_n \subseteq M$ . The treatment of the *operational* (that is: structural) part of the modalities remains to be looked at, of course.

## 5 Future work

Logically minded readers will have noticed that all that this paper really does is give an indication of how the problem of extending proof nets to modal types can be tackled. It has yet to be proved that a sequent is derivable iff a proof net can be constructed for it. In other words, soundness and completeness proofs must follow.

The next step would be the verification of the more speculative claims made about the extendability to structural modalities and polymodal systems. This would involve research into specialized string unification algorithms and, again, soundness and completeness proofs.

Finally, an actual parser (or, in logical terms: a theorem prover) could be programmed using the proof net approach. Such a parser could use a partially executed lexicon, as mentioned in section 3.

There remain many linguistic phenomena to be covered, even by the language extended with modalities. But, modalities bring us a lot closer to the goal at hand: an adequate parser that uses the same techniques as reasoning does: logic.

## References

- [1] Van Benthem, J. (1986) 'Categorial Grammar' Chapter 7 in *Essays in Logical Semantics*, Reidel, Dordrecht, pp.123-150.
- [2] Gabbay,D. (1991) *Labelled Deductive Systems* Draft. Oxford University Press (to appear)
- [3] Hendriks,H. and D.Roorda (1991) 'Spurious Ambiguity in Categorial Grammar', ITLI, Amsterdam.
- [4] Hepple,M. (1990) 'Efficient Lambek Theorem Proving', Chapter 6 in *The Grammar and Processing of Order and Dependency. A Categorial Approach*. Ph.D. Dissertation, University of Edinburgh.
- [5] Lambek, J. (1958) 'The Mathematics of Sentence Structure', *American Mathematical Monthly* 65, pp. 154-170.
- [6] Moortgat,M. (1988) *Categorial Investigations: Logical and Linguistic Aspects of the Lambek Calculus* Foris Publications, Dordrecht.
- [7] Moortgat,M. (1990) 'Categorial logics: a computational perspective'. In van de Goor (ed.) *Proceedings Computing Science in the Netherlands 1990* Mathematisch Centrum, Amsterdam, pp. 329-347.
- [8] Moortgat,M. (1991) 'Labelled Deductive Systems for categorial theorem proving' *Proceedings of the 8th Amsterdam Colloquium* (to appear).
- [9] Moortgat,M. and G. Morrill (1991) 'Heads and phrases. Type calculus for dependency and constituent structure'. OTS Working Papers,RUU,Utrecht.

- [10] Morrill, G. (1990) 'Intensionality and boundedness' *Linguistics and Philosophy*.
- [11] Morrill, G., N. Leslie, M. Hepple and G. Barry (1990) 'Categorical Deductions and Structural Operations' In Barry and Morrill (eds.) *Studies in Categorical Grammar*, CCS, Edinburgh.
- [12] Roorda, D. (1991) *Resource Logics. Proof-theoretical Investigations* Ph.D. Dissertation, Amsterdam.
- [13] Wallen, L.A. (1990) *Automated Deduction in Nonclassical Logics* MIT Press, Cambridge, London.