Geometry of Language

Glyn Morrill
Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Mòdul C5 - Campus Nord
Jordi Girona Salgado, 1--3
E-08034 Barcelona

E-mail: morrill@lsi.upc.es
HTTP: //www-lsi.upc.es/~glyn/

11th October 1997

Abstract

Girard (1987) introduced proof nets as a syntax of linear proofs which eliminates inessential rule ordering manifested by sequent calculus. Proof nets adapted to the Lambek calculus (Roorda 1991) fulfill a role in categorial grammar analogous to that of phrase structure trees in CFG so that categorial proof nets have a central part to play in computational syntax and semantics; in particular they allow a reinterpretation of the "problem" of spurious ambiguity as an *opportunity* for parallelism. This article aims to make three contributions: i) provide a tutorial overview of categorial proof nets, ii) apply and provide motivation for proof nets by showing how a partial execution eschews the need for semantic evaluation in language processing, and iii) analyse the intrinsic geometry of partially commutative proof nets for the kinds of discontinuity attested in language, offering proof nets for the in situ binder type-constructor $Q(\cdot, \cdot, \cdot)$ of Moortgat (1991/6).

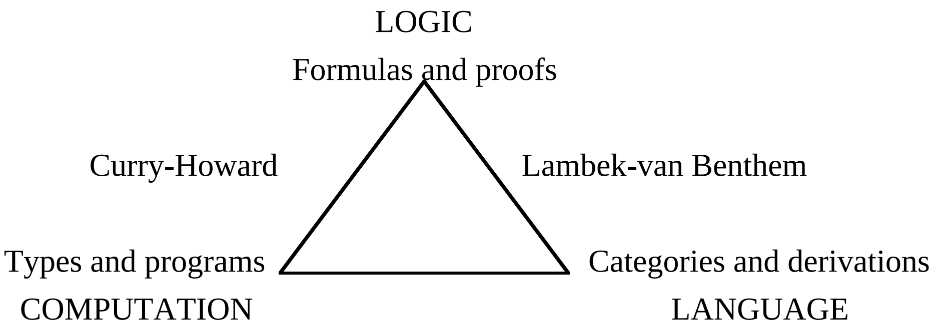
'Thus Riemann made the first momentous break with Newton in 200 years, banishing the action-as-a-distance principle. To Riemann, "force" was a consequence of geometry'

Kaku (1994, p.36)

0. Introduction

A logical model of computation can be founded on the *Curry-Howard correspondence*: *formulas-as-types* and *proofs-as-programs*. On this scheme logical foundations of computation are developed with computation as proof normalisation or Cut-elimination. We believe that in a similar way a logical model of language can be founded on what might be termed a '*Lambek-van Benthem correspondence*': *formulas-as-categories* and *proofs-as-derivations*:

(0)



This paper is concerned with developing such logical foundations of language. Categorial grammar predates both the Chomskian programme of formal syntax and that of Montague in formal semantics, but it is convenient to see the formulation of categorial grammar employed here, *Type Logical Grammar* (TLG), as the product of three revolutions in an integrated programme of syntax and semantics.

The first is a *linguistic* revolution of *lexicalism* (Bresnan 1982). This dismissed the notion that the lexicon should be the locus of only idiosyncratic information and that all regularities should be expressed as syntactic rules. Instead, linguistic generalisations could be expressed as regularities of lexical categorization. Classical transformational grammar had a *multistratal* architecture in which successive application of transformations related multiple levels of representation. The lexical relocation of information reduced this to a *bistratal* architecture relating just two levels of derivational representation, a level of constituent structure and a level of functional structure.

The second is a *computational* revolution of *compositionality* (Gazdar *et al*. 1985). This incorporated a Montagovian notion of compositionality whereby the syntactic rule applications in a phrase structure are correlated with semantic operations. Consequently, linguistic dependencies are localised in a single level of derivational representation in a computationally amenable *monostratal* architecture.

The third is a *logical* revolution of *minimalism* in which *formal* grammatical systems are refined to *logical* ones. By a logical system we mean a formal language with an interpretation which defines consequence in terms of a model-theoretic notion of validity. Any formal system can be embedded in a logic by representing its rules as axioms; we mean a refinement of grammar to logic *without non-logical axioms*. In this case there is a *nostratal* architecture with no essential level of derivational representation: all properties are projected from the model-theoretic specification of what the formalism signifies. We take this to be the essential nature of TLG (Moortgat 1988, 1997; van Benthem 1991/95; Morrill 1994; Carpenter 1997).

A language, on the Saussurian view, is a collection of signs, where each sign associates a signifier and a signified. A grammar, as a description of language, is to specify a set of pairings of representations of signifiers and signifieds (we shall say prosodic and semantic forms) corresponding to the signs. On such a view syntax is not in the language: note that there is no observation which bears specifically on syntax in the way that there are observations which bear specifically on prosodics and semantics. The nostratal TLG architecture simply retains an agnosticism regarding an issue on which the Saussurian view makes no demand.

Another outlook on language is that it is fundamentally dynamic: that the basic linguistic phenomena are productions and comprehensions of signs, that is, computational processes. To build such a view on top of a static Saussurian perspective one becomes concerned with the *calculation* of semantics from prosodics and of prosodics from semantics, that is with proof theory of categorial logic. But proofs can be presented in any number of formats, and to escape notational irrelevancies we want to identify the pure structure of proofs.

To the question "What is the pure, geometric, structure of a proof?" there is quite a good answer in the case of say implicational intuitionistic logic: it is the structure of Prawitz-style natural deductions or (what is the same thing) simply typed lambda terms. For the linear logic of Girard (1987) there is a rather different answer: the geometry of proofs is given by the notion of *proof net*. For categorial logic, being akin to linear logic, the answer is expected to be some variety of the same. Thus, just as the logical modelling of computation yields the outlook *computation as Cut-elimination*, we wish to propose here that logical modelling of language yields the outlook *syntactic structures as proof nets*.

In sections 1--8 we present a tutorial overview of categorial proof nets. In section 9 we apply and provide motivation for proof nets by showing how a partial execution eschews the need for semantic evaluation in language processing. In section 10 we analyse the intrinsic geometry of partially commutative proof nets for the kinds of discontinuity attested in language, offering in particular proof nets for the in situ binder type-constructor $Q(\cdot, \cdot, \cdot)$ of Moortgat (1991/6).

1. Paraphrase, long distance extraction, quantificational ambiguity, and in situ binding

We will formulate our observations by reference to paraphrase, long distance extraction, quantificational ambiguity, and in situ binding. Paraphrase is exemplified by the following:

(1) a. Frodo inhabits Bag End.
    b. Frodo lives in Bag End.

Both of these sentences have a logical semantic form which can be represented by the following functional term, where $(\phi\ \psi)$ is the functional application of ϕ to ψ:

(2) ((*in b*) (*live f*))

Likewise, the sentences (3a) and (3b) are paraphrases, sharing a semantic form (3c).

(3) a. John tries to find Mary.
    b. John seeks Mary.
    c. ((*try* (*find m*)) *j*)

Long distance extraction, as realised in such constructions as relativization, interrogativization and topicalization, is illustrated in (4).

(4) a. (John met the) man whom$_i$ Mary saw the brother of e_i
    b. (John met the) man whom$_i$ Bill said Mary saw the brother of e_i
    c. (John met the) man whom$_i$ Suzy thinks Bill said Mary saw the brother of e_i

Sometimes an extracted element can bring with it part of its context (“pied piping”):

(5) (John met the) man [the brother of whom]$_i$ Mary saw e_i

Example (5) is a paraphrase of (4a). The semantic form of both can be represented by the functional term (6) where $\lambda x\phi$ is the functional abstraction of ϕ over x. The long distance dependency between the relative pronoun and the extraction site is mediated by variable binding.

(6) $((rel\ \lambda x((see\ (the\ (brother\ (of\ x))))\ m))\ man)$

Example (7a) is also a paraphrase of (4a), but (7b) is ungrammatical: ‘that’ does not exhibit pied piping.

(7) a. (John met the) man that$_i$ Mary saw the brother of e_i
b. *(John met the) man [the brother of that]$_i$ Mary saw e_i

Pied piping is a form of in situ binding: in (5) ‘whom’ converts its context ‘the brother of ...’ into a relativising element. Linguistically, we want to categorize ‘whom’ and ‘that’ in such a way that both realize long distance extraction but only the former realizes pied piping.

Another form of in situ binding is manifested by quantification. Quantifiers occur embedded in sentences, but semantically they take sentential scope:

(8) a. John gave someone Fido.
b. $(someone\ \lambda x(((give\ x)\ f)\ j))$

Sometimes quantification gives rise to ambiguity. One kind of quantificational ambiguity is the differentiation of *de re* and *de dicto* readings in a case such as (9) where a quantifier occurs within the scope of a propositional attitude verb.

(9) John believes someone runs.

On the *de re* (or: specific) interpretation it is reported that there is someone in the world of evaluation of the overall report towards whom John holds a belief:

(10) $(someone\ \lambda x((believe\ (run\ x))\ j))$

On the *de dicto* (or: non-specific) interpretation it is just reported that in the world of John's beliefs there is held to be someone who runs:

(11) $((believe\ (someone\ \lambda x(run\ x)))\ j)$

A similar ambiguity arises in an example such as (12).

(12) John tries to find someone.

There is a *de re*/specific reading in which it is reported that there is some specific person in the world of evaluation towards whom John's efforts are directed, and a *de dicto*/non-specific one in which it is just reported that John aspires to achieve a state wherein some or other person has been found. The paraphrase (13) exhibits exactly the same two readings.

(13) John seeks someone.

Another kind of quantificational ambiguity is that between subject wide scope and object wide scope readings in cases like (14a). In the subject wide scope interpretation (14b) it is only required that each person loves some, in general different, person. In the object wide scope interpretation (14c) it is required that there be a single person loved by all.

(14) a. Everyone loves someone.
b. $(everyone\ \lambda x(someone\ \lambda y((love\ y)\ x)))$
c. $(someone\ \lambda y(everyone\ \lambda x((love\ y)\ x)))$

Note that (14c) entails (14b), but not vice versa.

2. Categorial grammar

We consider categorial grammar with types (or: formulas, or: categories) defined by the following grammar:

(15) a. $F ::= A \mid F\backslash F \mid F/F \mid F\bullet F$
b. $A ::= \mathrm{S} \mid \mathrm{N} \mid \mathrm{CN} \mid \mathrm{PP} \mid \ldots$

The types in A are referred to as atoms (or: primitives) and correspond to the kinds of expressions which are considered to be “complete”. Fairly uncontroversially, this class may be taken to include at least sentences and names; what the class is is not fixed by the formalism.

The left division type $A\backslash B$ ('A under B') is that of expressions (functors) which concatenate with (arguments) in A on the left to yield Bs. The right division type B/A ('B over A') is that of expressions (functors) which concatenate with (arguments) in A on the right yielding Bs. The product type $A\bullet B$ is that of expressions which are the result of concatenating an A with a B; products do not play a dominant role here.

More precisely, let L be the set of strings (including the empty string ε) over a finite vocabulary V and let + be the operation of concatenation (i.e. $(L, \varepsilon, +)$ is the free monoid generated by V)[1]. Each type A is interpreted as a subset $D(A)$ of L. When the interpretation of atomic types has been fixed, that of complex types is defined by (16).

(16)
$$[[A\backslash B]] = \{s|\ \forall s'\in [[A]],\ s'+s\in [[B]]\}$$
$$[[B/A]] = \{s|\ \forall s'\in [[A]],\ s+s'\in [[B]]\}$$
$$[[A\bullet B]] = \{s_1+s_2|\ s_1\in [[A]]\ \&\ s_2\in [[B]]\}$$

The following are some examples of types:

(17)

S	sentence	John runs, Mary gives John Fido
N	name	John, the man
CN	common noun	man, man that John sees
PP	prepositional phrase	of John
PP/N	preposition	of
N\S	intransitive verb	runs, finds Mary
(N\S)/N	transitive verb	finds, gives John
((N\S)/N)/N	ditransitive verb	gives
(N\S)/S		says, believes
(N\S)/(N\S)		tries

In general, given some type assignments others may be inferred. Such reasoning is precisely formulated in the Lambek calculus **L**.

[1] In fact Lambek (1958) excluded the empty string ---and hence empty antecedents in the calculus of (18)--- but it is convenient to include it here.

3. Lambek sequent calculus

In the sequent calculus of Lambek (1958) a *sequent* $\Gamma \Rightarrow A$ consists of a sequence Γ of types (the antecedent) and a type A (the succedent). A sequent states that the concatenation of expressions of the types in Γ yields an expression of the type A. The valid sequents are the theorems derivable from the following axiom and rule schemata.[2]

(18) a.

$$\frac{}{A \Rightarrow A}\ \mathrm{id} \qquad\qquad \frac{\Gamma \Rightarrow A \qquad \Delta1, A, \Delta2 \Rightarrow B}{\Delta1, \Gamma, \Delta2 \Rightarrow B}\ \mathrm{Cut}$$

b.

$$\frac{A, \Gamma \Rightarrow B}{\Gamma \Rightarrow A\backslash B}\ \backslash\mathrm{R} \qquad\qquad \frac{\Gamma \Rightarrow A \qquad \Delta1, B, \Delta2 \Rightarrow C}{\Delta1, \Gamma,\ A\backslash B, \Delta2 \Rightarrow C}\ \backslash\mathrm{L}$$

c.

$$\frac{\Gamma, A \Rightarrow B}{\Gamma \Rightarrow B/A}\ /\mathrm{R} \qquad\qquad \frac{\Gamma \Rightarrow A \qquad \Delta1, B, \Delta2 \Rightarrow C}{\Delta1, B/A, \Gamma, \Delta2 \Rightarrow C}\ /\mathrm{L}$$

d.

$$\frac{\Gamma1 \Rightarrow A \qquad \Gamma2 \Rightarrow B}{\Gamma1, \Gamma2 \Rightarrow A\bullet B}\ \bullet\mathrm{R} \qquad\qquad \frac{\Gamma1, A, B, \Gamma2 \Rightarrow C}{\Gamma1, A\bullet B, \Gamma2 \Rightarrow C}\ \bullet\mathrm{L}$$

The calculus **L** lacks the usual structural rules of permutation, contraction and weakening. Adding permutation collapses the two divisions into a non-directional implication $-\!\circ$ and yields the multiplicative fragment of intuitionistic linear logic, known as the Lambek-van Benthem calculus **LP**.[3]

The validity of the id axiom and the Cut rule follows from the reflexivity and the transitivity respectively of set containment. The calculus enjoys the property of *Cut elimination* whereby every proof has a Cut-free equivalent (indeed, one in which only atomic id axioms are used: what we shall call $\beta\eta$-long sequent proofs).[4] Thus,

---

[2]The completeness of the calculus with respect to the intended interpretation was proved in Pentus (1993).

[3]Adding also contraction and weakening we obtain the implicational and conjunctive fragment of intuitionistic logic. Thus every Lambek proof can be read as an intuitionistic proof and has a constructive content which can be identified with its intuitionistic normal form natural deduction proof (Prawitz 1965) or, what is the same thing under the Curry-Howard correspondence, its normal form as a typed lambda term.

[4]By ‘equivalent’ we mean a proof of the same theorem with the same constructive content (fn. 3).

processing can be performed using just the left (L) and right (R) rules, which introduce connectives in the left and the right of the conclusion sequents respectively. These rules all have exactly one connective occurrence less in the premises than in the conclusion, therefore one can compute all the (Cut-free) proofs of any sequent by traversing the finite space of proof search without Cut.

By way of illustration of the sequent calculus, the following is a proof of a theorem of lifting, or (subject) type raising:

(19)

$$\dfrac{\dfrac{\mathrm{N} \Rightarrow \mathrm{N} \qquad \mathrm{S} \Rightarrow \mathrm{S}}{\mathrm{N}, \mathrm{N}\backslash\mathrm{S} \Rightarrow \mathrm{S}}\,\backslash L}{\mathrm{N} \Rightarrow \mathrm{S}/(\mathrm{N}\backslash\mathrm{S})}\,/R$$

Where *a* labels the antecedent, the coding of this proof as a lambda term ---what we shall call the derivational semantics--- is $\lambda x(x\ a)$.). The converse of lifting, lowering, in (20a) is not derivable. A proof of a theorem of composition (it has as its semantics functional composition) is given in (20b).

(20) a. $\mathrm{S}/(\mathrm{N}\backslash\mathrm{S}) \Rightarrow \mathrm{N}$

b.

$$\dfrac{\dfrac{A \Rightarrow A \qquad \dfrac{B \Rightarrow B \qquad C \Rightarrow C}{B, B\backslash C \Rightarrow C}\,\backslash L}{A, A\backslash B, B\backslash C \Rightarrow C}\,\backslash L}{A\backslash B, B\backslash C \Rightarrow A\backslash C}\,\backslash R$$

A grammar contains a set of lexical assignments α: A of types to expressions; the lexical expressions, which may or may not comprise exactly one word, may receive multiple assignments (lexical ambiguity). An expression $w_1+\ldots+w_m$ is of type A just in case $w_1+\ldots+w_m$ is the concatenation $\alpha_1+\ldots+\alpha_n$ of lexical expressions such that α_i: A_i, $1\leq i\leq n$, and $A_1, \ldots, A_n \Rightarrow A$ is valid. For instance, assuming the expected lexical type assignments to proper names and intransitive, transitive and ditransitive verbs, there are the following derivations:

(21)

$$\dfrac{\dfrac{N \Rightarrow N \qquad S \Rightarrow S}{N, N\backslash S \Rightarrow S}\,\backslash L}{\mathbf{john}+\mathbf{runs}: S}$$

(22)

$$\dfrac{\dfrac{N \Rightarrow N \qquad \dfrac{N \Rightarrow N \qquad S \Rightarrow S}{N, N\backslash S \Rightarrow S}\,\backslash L}{N, (N\backslash S)/N, N \Rightarrow S}\,/L}{\mathbf{john}+\mathbf{finds}+\mathbf{mary}: S}$$

(23)

$$\dfrac{\dfrac{N \Rightarrow N \qquad \dfrac{N \Rightarrow N \qquad \dfrac{N \Rightarrow N \qquad S \Rightarrow S}{N, N\backslash S \Rightarrow S}\,\backslash L}{N, (N\backslash S)/N, N \Rightarrow S}\,/L}{N, ((N\backslash S)/N)/N, N, N \Rightarrow S}\,/L}{\mathbf{john}+\mathbf{gives}+\mathbf{mary}+\mathbf{fido}: S}$$

Ungrammaticality occurs when there is invalidity of the sequents arising by lexical insertion, as in the following:

(24)

$$\dfrac{N\backslash S, N \Rightarrow S}{\mathbf{runs}+\mathbf{john}: S}$$

4. Ambiguity and spurious ambiguity

The sentence (25) is structurally ambiguous.

(25) Sometimes it rains surprisingly.

There is a reading “it is surprising that sometimes it rains” and another “sometimes the manner in which it rains is surprising”. As would be expected there are in such a case distinct derivations corresponding to alternative scopings of the adverbials:

(26) a.

$$\frac{S/S,\ S,\ S\backslash S \Rightarrow S}{\textbf{sometimes}+\textbf{it}+\textbf{rains}+\textbf{surprisingly}\text{: } S}$$

b.

$$\frac{S \Rightarrow S \qquad \dfrac{S \Rightarrow S \qquad S \Rightarrow S}{S/S,\ S \Rightarrow S}\,/L}{S/S,\ S,\ S\backslash S \Rightarrow S}\,\backslash L$$

c.

$$\frac{S \Rightarrow S \qquad \dfrac{S \Rightarrow S \qquad S \Rightarrow S}{S,\ S\backslash S \Rightarrow S}\,\backslash L}{S/S,\ S,\ S\backslash S \Rightarrow S}\,/L$$

However, sometimes a non-ambiguous expression also has more than one sequent proof (even excluding Cut); thus the sequent in (27a) has the proofs (27b) and (27c).

(27) a.

$$\frac{N/CN,\ CN,\ N\backslash S \Rightarrow S}{\textbf{the}+\textbf{man}+\textbf{runs}\text{: } S}$$

b.

$$\frac{CN \Rightarrow CN \qquad \dfrac{N \Rightarrow N \qquad S \Rightarrow S}{N,\ N\backslash S \Rightarrow S}\,\backslash L}{N/CN,\ CN,\ N\backslash S \Rightarrow S}\,/L$$

c.

$$\frac{\dfrac{CN \Rightarrow CN \qquad N \Rightarrow N}{N/CN,\ CN \Rightarrow N}\,/L \qquad S \Rightarrow S}{N/CN,\ CN,\ N\backslash S \Rightarrow S}\,\backslash L$$

As the reader may check, N/CN, CN $\Rightarrow$ S/(N\S) has three Cut-free proofs; in general the combinatorial possibilities multiply exponentially. This feature is sometimes referred to as the problem of spurious ambiguity or derivational equivalence. It is regarded as problematic computationally because it means that in an exhaustive traversal of the proof search space one must either repeat subcomputations, or else perform book-keeping to avoid so doing.

The problem is that different βη-long sequent derivations do not necessarily represent different readings, and this is the case because the sequent calculus forces us to choose between a sequencialisation of inferences ---in the case of (27) instances of /L and \L--- when in fact they are not ordered by dependency and can be performed in parallel.

The problem can be resolved by defining stricter normalised proofs which impose a unique ordering when alternatives would otherwise be available (König 1990, Hepple 1991, Hendriks 1993). However, while this removes spurious ambiguity as a problem arising from independence of inferences, it signally fails to exploit the fact that such inferences can be parallelised. Thus we prefer the term ‘derivational equivalence’ to ‘spurious ambiguity’ and interpret the phenomenon not as a problem for sequencialisation, but as an opportunity for parallelism. This opportunity is grasped in *proof nets*.

5. Proof nets for **L**

Proof nets for **L** were first developed in detail by Roorda (1991), adapting their original introduction for linear logic in Girard (1987). In proof nets, the opposition of types arising from their location in either the antecedent or the succedent of sequents is replaced by assignment of negative (antecedent) or positive (succedent) polarity. A proof net here is a connected graph of polar types.

First we define a more general concept of *proof structure*. These are graphs assembled out of the following components:

(28) a.

X ~*X*	*X* ~*X*
id:	Cut:
zero premises,	two premises,
two conclusions	zero conclusions

b.

$$
\begin{array}{ccc}
B+ & & A- \\
\diagdown & \text{i} & \diagup \\
& A\backslash B+ &
\end{array}
\qquad\qquad
\begin{array}{ccc}
A+ & & B- \\
\diagdown & \text{ii} & \diagup \\
& A\backslash B- &
\end{array}
$$

$$
\begin{array}{ccc}
A- & & B+ \\
\diagdown & \text{i} & \diagup \\
& B/A+ &
\end{array}
\qquad\qquad
\begin{array}{ccc}
B- & & A+ \\
\diagdown & \text{ii} & \diagup \\
& B/A- &
\end{array}
$$

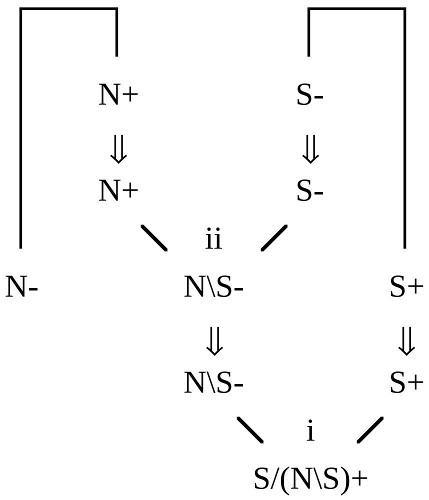
$$
\begin{array}{ccc}
B+ & & A+ \\
\diagdown & \text{ii} & \diagup \\
& A\bullet B+ &
\end{array}
\qquad\qquad
\begin{array}{ccc}
A- & & B- \\
\diagdown & \text{i} & \diagup \\
& A\bullet B- &
\end{array}
$$

i- and ii-cells: two premises, one conclusion

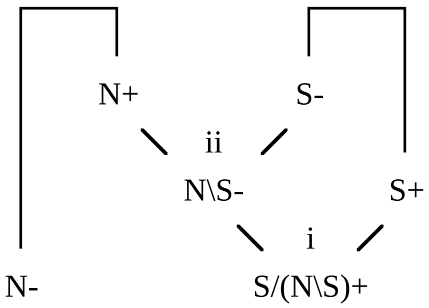
In the id and Cut components X and $\sim X$ schematise over occurrences of the same type with opposite polarity (either way round). These are sometimes referred to as id and Cut *links*. Note that the nodes of components are also marked (implicitly) as being either conclusions (looking down) or premises (looking up). In the i- and ii-cells the middle nodes are the conclusions and the outer nodes the premises. The i-cells correspond to unary sequent rules and the ii-cells to binary sequent rules. Observe that in the positive (succedent), but not in the negative (antecedent) unfoldings, the order of subformulas is switched between premises and conclusion.

The proof structures are assembled by identifying nodes of the same polar type which are the premises and conclusions of different components; premises and conclusions not unified in this way are the premises and conclusions of the proof structure as a whole. For example, in (29a) two id components and two cells are assembled into a proof structure (29b) with no premises and two conclusions, N- and S/(N\S)+:

(29) a.



b.



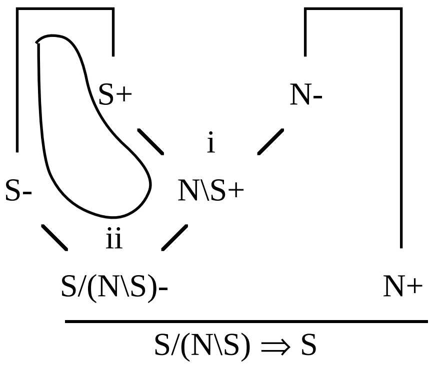
The proof structures arise, essentially, by forgetting the contexts (Γ and Δ) of the sequent rules, and not all proof structures are well-formed as proofs. There must exist a global synchronization of the partitioning of contexts by rules (the long trip condition). We shall say that a "circularity" is a circuit which is elementary (i.e. it only traverses edges once) and which does not traverse both edges of any i-cell. A proof structure is well-formed, a *module* (partial proof net), only if it contains no circularity. A module is a *proof net* only if it contains no premises. The structure (29b) is a proof net, in fact it is the proof net for our instance (19) of lifting since its conclusions are the polar types for this sequent:

(30)

$$\frac{\mathrm{N}- \qquad\qquad\qquad \mathrm{S/(N\backslash S)+}}{\mathrm{N} \Rightarrow \mathrm{S/(N\backslash S)}}$$

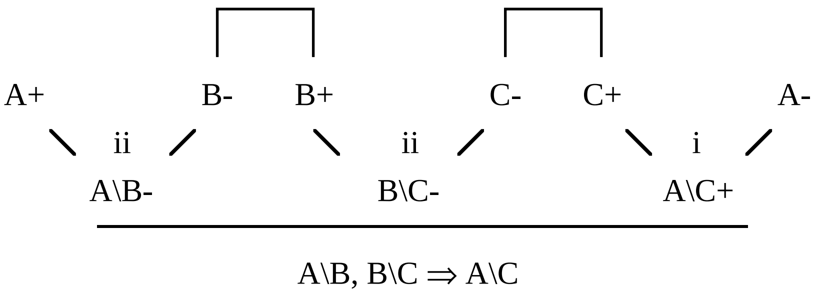
The structure in (31) is not a module because it contains the circularity indicated: it corresponds to the lowering (20a), which is invalid.

(31)



The structure (32) is a module with two premises and three conclusions; the latter are the polar types of our composition theorem (20b). Adding the remaining id axiom link makes it a proof net for composition.

(32)

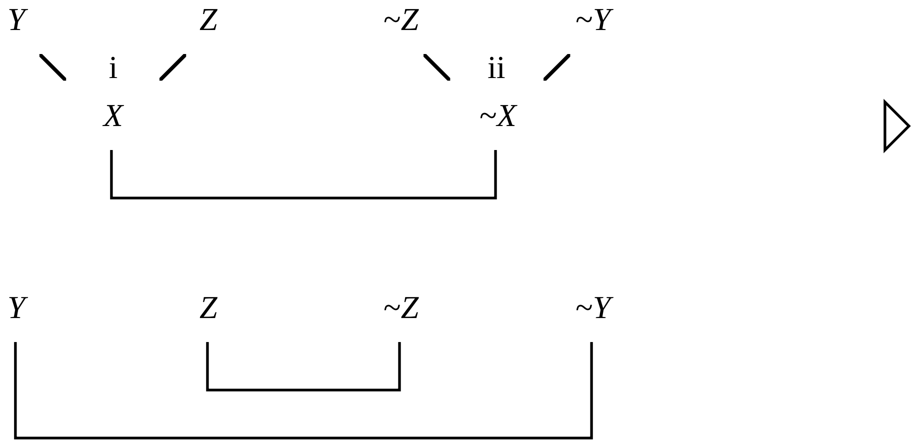


For **L** proof nets must be *planar*, i.e. with no crossing edges. This corresponds to the non-commutativity of **L**. In **LP**, linear logic, which is commutative, there is no such requirement.

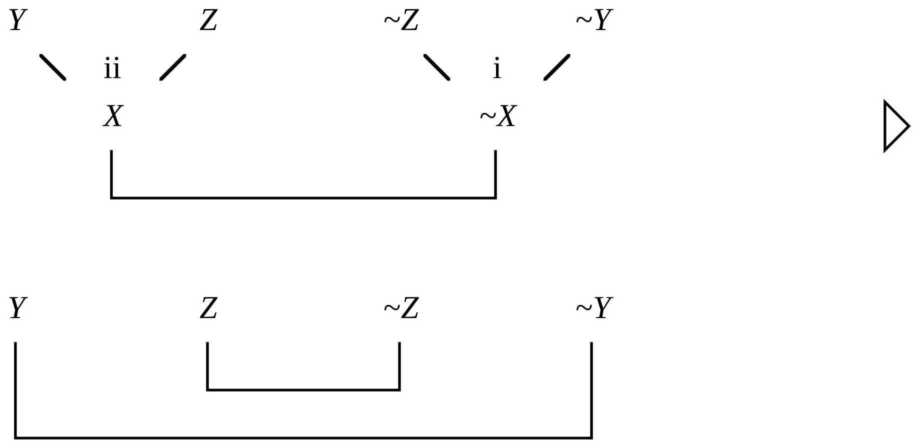
6. Cut eliminacion

Proof nets, like the sequent calculus, enjoy the Cut elimination property whereby every proof has a Cut-free equivalent. The evaluation of a net to its Cut-free normal form is a process of graph reduction. The reductions are as follows:

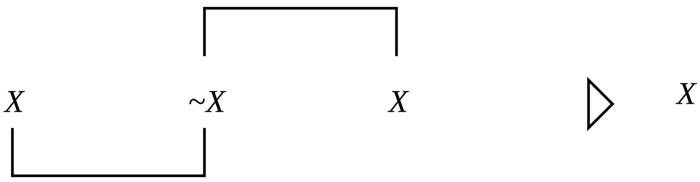
(33) a.



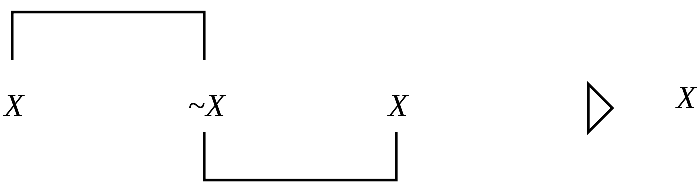
b.



(34) a.



b.



We shall see examples shortly in the context of the application of section 9.

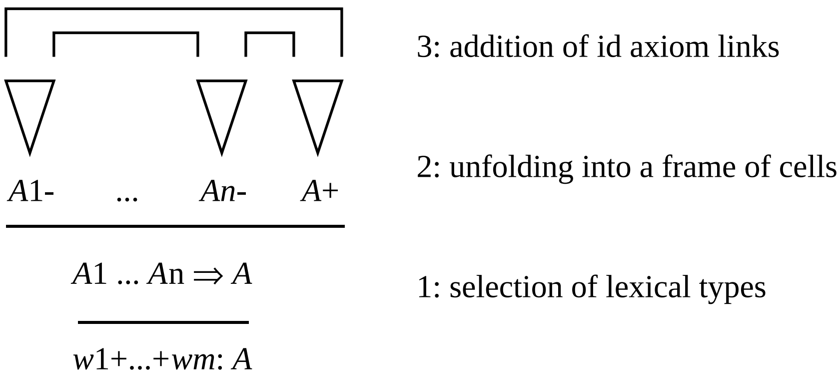
7. Automated language processing

As is the case for the sequent calculus, with proof nets every proof has a Cut-free equivalent in which only atomic id axiom links are used: what we shall call βη-long proof nets. However, whereas some βη-long sequent proofs are equivalent, leading to spurious ambiguity/derivational equivalence, distinct βη-long proof nets always have distinct readings.

Performing search for βη-long proof nets resolves the problem of spurious ambiguity and at the same time represents parallelism which can be simulated by memoisation or, in computational linguistic terms, chart parsing (Morrill 1996).

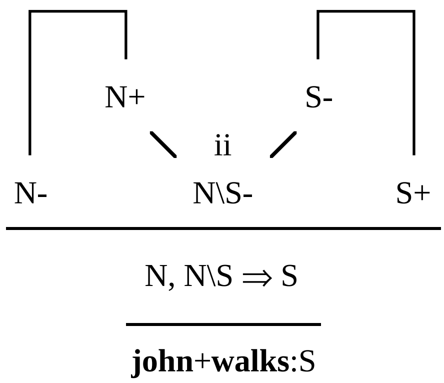
The analysis of an expression as search for proof nets can be construed in three phases, 1) selection of lexical types for elements in the expression, 2) unfolding of these types into a *frame* of trees of i- and ii-cells with atomic leaves (literals), and 3) addition of (planar) id axiom links to form proof nets:

(35)



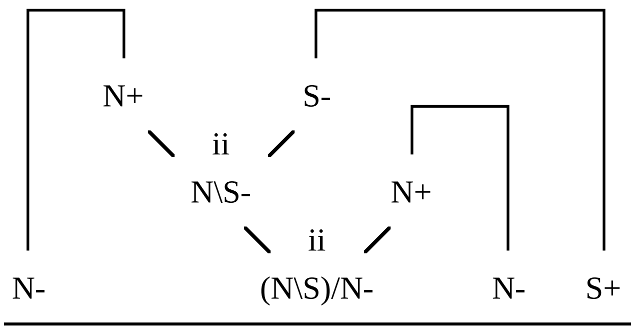
For example, ‘John walks’ has the following analysis:

(36)



The sentence ‘John finds Mary’ has the analysis given in (37).

(37)

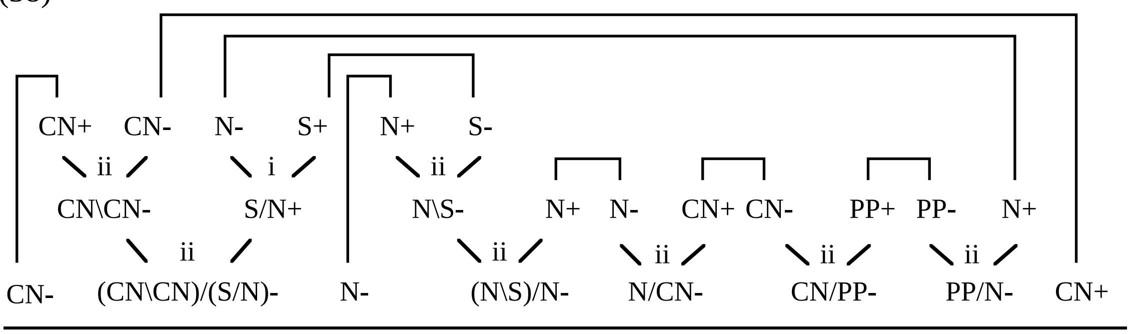


N, (N\S)/N, N $\Rightarrow$ S

john+**finds**+**mary**: S

Long distance extraction as exemplified by (7a) has the analysis (38); the relative pronoun has a higher order type (Steedman 1985) and other types are as would be expected.

(38)

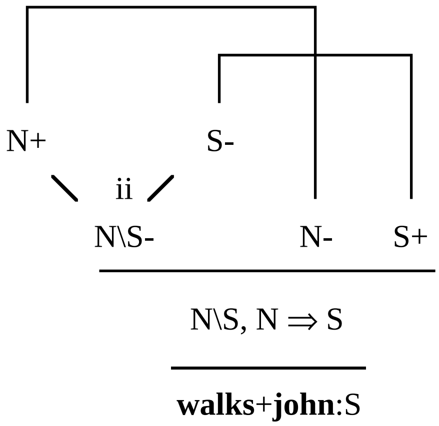


CN, (CN\CN)/(S/N), N, (N\S)/N, N/CN, CN/PP, PP/N $\Rightarrow$ CN

man+**that**+**mary**+**saw**+**the**+**brother**+**of**: CN

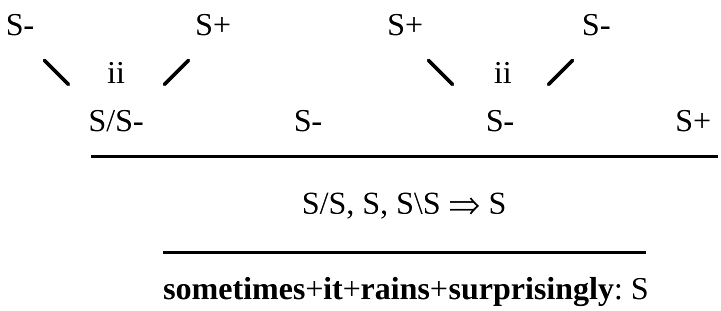
The ungrammaticality of ‘walks John’ is attested by the non-planarity of the proof structure (39).

(39)

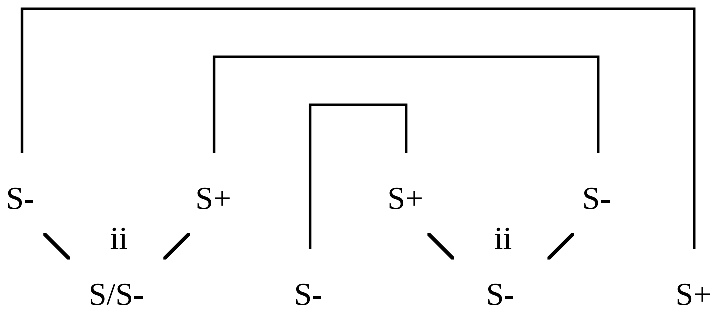


As expected, where there is structural ambiguity there are multiple derivations:

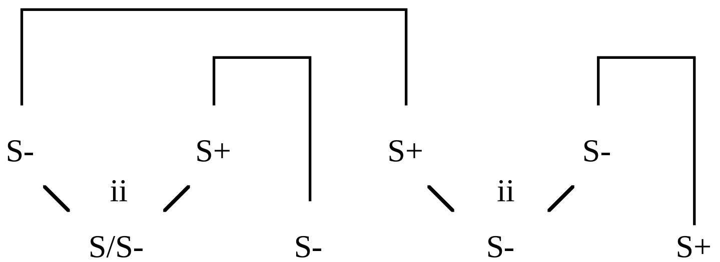
(40) a.



b.

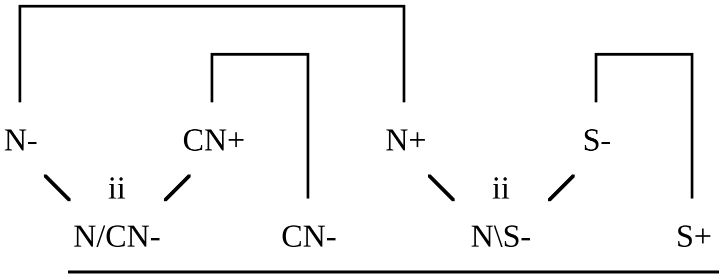


c.



But now also, when there is not structural ambiguity there is only one derivation:

(41)



$$\frac{N/CN,\ CN,\ N\backslash S \Rightarrow S}{\textbf{the}+\textbf{man}+\textbf{runs}\text{: } S}$$

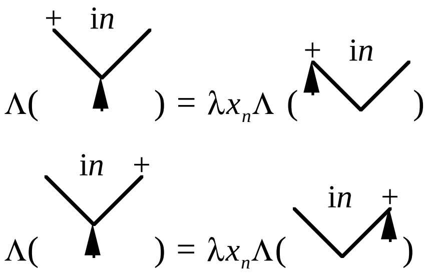
This property is entirely general: the problem of spurious ambiguity is resolved.

8. How to extract the semantic content from a net

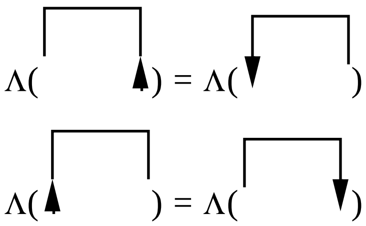
Until now we have not been explicit about how a proof determines a semantic reading. We shall show here how to extract from a proof net a functional term representing the semantics (see de Groote and Retoré 1996, who reference Lamarche 1995). This is done by travelling from premises to conclusions and from conclusions to premises in a proof net following deterministic instructions. The proof nets are proof structures in which following these instructions visits each node exactly once.

First one assigns a distinct index to each i-cell; then one starts travelling upwards through the unique positive conclusion. Thereafter one proceeds as follows (for brevity we exclude product):

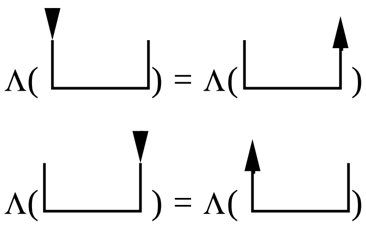
(42) a. Going up through the conclusion of a i-cell, make a functional abstraction and continue upwards through the positive premise:



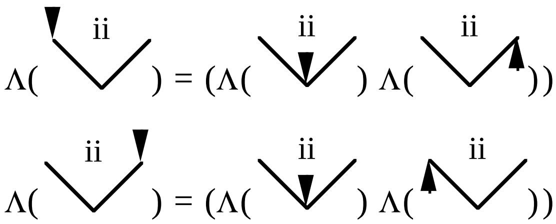
b. Going up through one id conclusion, go down through the other:



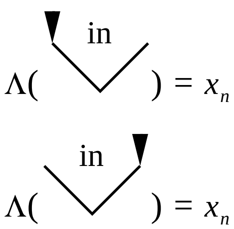
c. Going down through one premise of Cut, go up through the other:



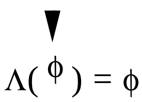
d. Going down through one premise of a ii-cell, make a functional application and continue going down through the conclusion (function) and going up through the other (argument):



e. Going down through the premise of a i-cell, put the corresponding bound variable:

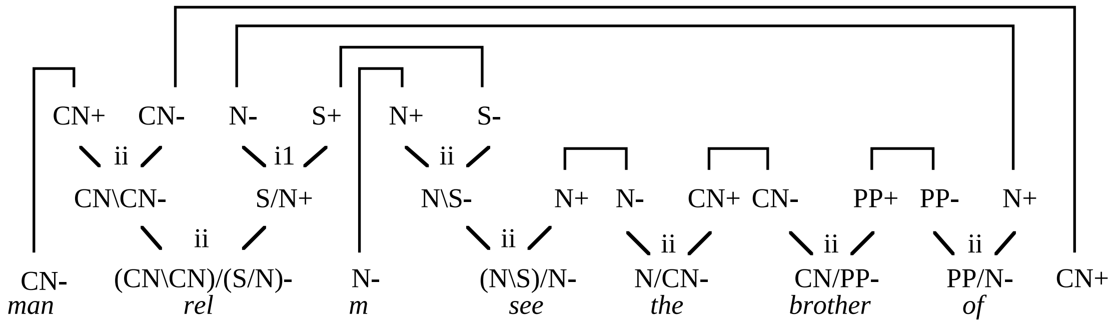


f. Going down through a terminal node, substitute the associated lexical semantics:



For example, traversal of the proof net (43a) generated in (38) for (7a) proceeds as shown in (43b), where * marks the point at construction and roman numerals indicate the argument traversals, performed after the function traversals, triggered by entry into ii-cells.

(43) a.



b. (* I)

((* II) I)

((*rel* *) I)

((*rel* λx_1*) I)

((*rel* λx_1(* III)) I)

((*rel* λx_1((* IV) III)) I)

((*rel* λx_1((*see* *) III)) I)

((*rel* λx_1((*see* (* V)) III)) I)

((*rel* λx_1((*see* (*the* *)) III)) I)

((*rel* λx_1((*see* (*the* (* VI))) III)) I)

((*rel* λx_1((*see* (*the* (*brother**))) III)) I)

((*rel* λx_1((*see* (*the* (*brother* (* VII)))) III)) I)

((*rel* λx_1((*see* (*the* (*brother* (*of* *)))) III)) I)

((*rel* λx_1((*see* (*the* (*brother* (*of* x_1)))) *)) I)

((*rel* λx_1((*see* (*the* (*brother* (*of* x_1)))) *m*)) *)

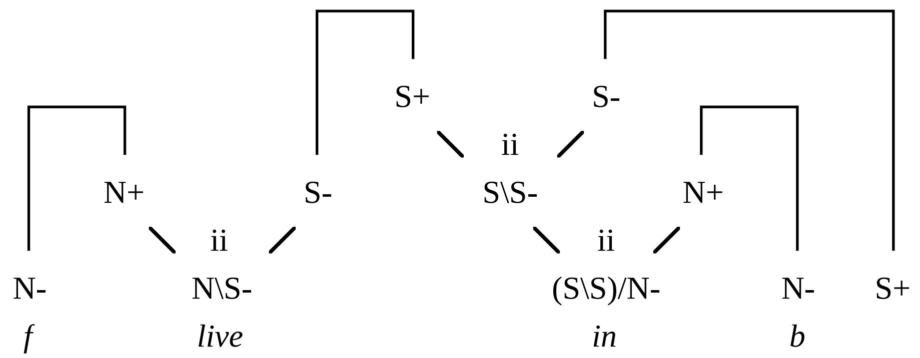
((*rel* λx_1((*see* (*the* (*brother* (*of* x_1)))) *m*)) *man*)

Let us observe that the following lexical type assignments capture the paraphrasing of (1a) and (1b); α-ϕ := A signifies the assignment to type A of expression α with lexical semantics ϕ.

(44)	a.	**frodo**	-	f
			:=	N
	b.	**lives**	-	*live*
			:=	N\S
	c.	**in**	-	*in*
			:=	(S\S)/N
	d.	**bag**+**end**	-	*b*
			:=	N
	e.	**inhabits**	-	$\lambda x \lambda y((in\ x)\ (live\ y))$
			:=	(N\S)/N

Then (1a) has the analysis (45a) with semantics (45b).

(45) a.



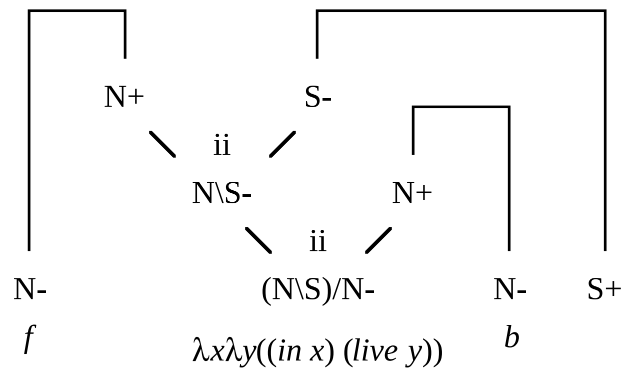
N, N\S, (S\S)/N, N ⇒ S

frodo+**lives**+**in**+**bag**+**end**: S

b. (* I)
((* II) I)
((*in* *) I)
((*in* *b*) *)
((*in* *b*) (* III))
((*in* *b*) (*live* *))
((*in* *b*) (*live* *f*))

Example (1b) has the analysis (46a) from which the semantics extracted is (46b).

(46) a.



N, (N\S)/N, N $\Rightarrow$ S

frodo+**inhabits**+**bag**+**end**: S

b. (* I)

((* II) I)

((λ*x*λ*y*((*in x*) (*live y*)) *) I)

((λ*x*λ*y*((*in x*) (*live y*)) *b*) *)

((λ*x*λ*y*((*in x*) (*live y*)) *b*) *f*)

This is not the same semantic term as that in (45b) but it reduces to the same by β-conversion, showing that the semantic content in the two cases is identical, that is, that there is paraphrase:

(46) ((λ*x*λ*y*((*in x*) (*live y*)) *b*) *f*) =

λ*y*((*in b*) (*live y*)) *f*) =

((*in b*) (*live f*))

Such λ-conversion only calculates what the grammar defines and is not part of the grammar itself, but computationally one might hope to aspire to on-line processing in which such manipulation is not necessary, for example by always maintaining normal semantic forms. This can be achieved with proof nets by partial evaluation in an off-line lexical compilation.

9. Partial evaluation of substitution of lexical semantics

In the processing as presented so far semantic evaluation is, as is usual, normalisation of the result of substituting lexical semantics into derivational semantics. Logically speaking, this substitution at the lexico-syntactic interface is a Cut, and the normalisation is Cut elimination. Currently the substitution and Cut elimination is executed after the proof search. However, if lexical semantics is represented as a proof net, one can calculate off-line the module resulting from connecting the lexical semantics with a Cut to the module resulting from the unfolding of the lexical types.[5]

A linear λ-term is one in which each abstraction binds exactly one variable occurrence. Lexical semantics expressed as a linear λ-term is unfolded into a proof net by the algorithm (47) (we do not consider the net ordered):

(47) a. Start with the λ-term ϕ at a + node: ϕ+.

b. To unfold $\lambda x_n\phi$+, make it the conclusion of a i-cell with index n and unfold ϕ+ at the positive premise:

ϕ+

in

$\lambda xn\phi$+

c. To unfold $\lambda x_n\phi$-, make it a Cut premise and unfold $\lambda x_n\phi$+ at the other premise:

$\lambda xn\phi$- $\lambda xn\phi$+

d. To unfold (ψ ϕ)-, make it the premise of a ii-cell and unfold ϕ+ at the conclusion and ψ- at the other premise:

(ϕ ψ)- ψ+

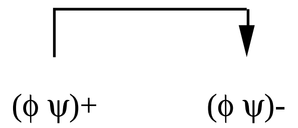
ii

ϕ-

---

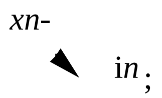
[5] The resulting association of modules with words is reminiscent of Lecomte and Retoré (1995) but their motivation is not made with reference to semantic processing, it being to make lexical categorization more expressive syntactically by using modules in general to classify words rather than just types (=modules without id or Cut links).

e. To unfold $(\psi\ \phi)$+ make it the conclusion of an id link and unfold $(\psi\ \phi)$- at the other conclusion:

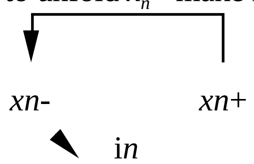


f. At a constant *k*- unfolding stops; to unfold a constant *k*+ make it an id premise first:

g. To unfold a bound variable x_n- make it the other premise of the i-cell with index *n*:

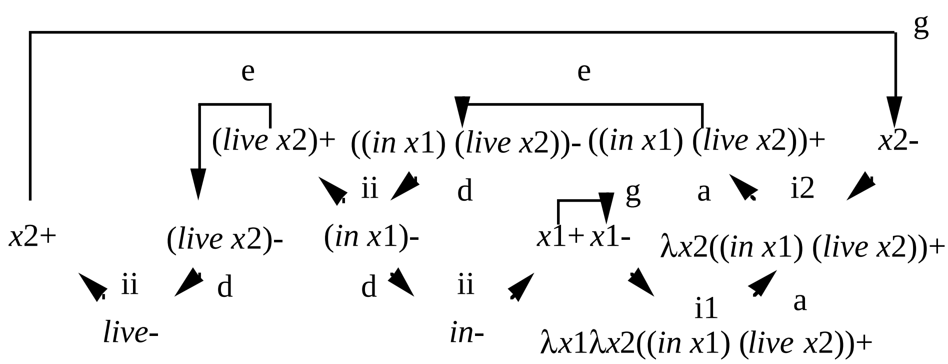


to unfold x_n+ make it an id premise first:



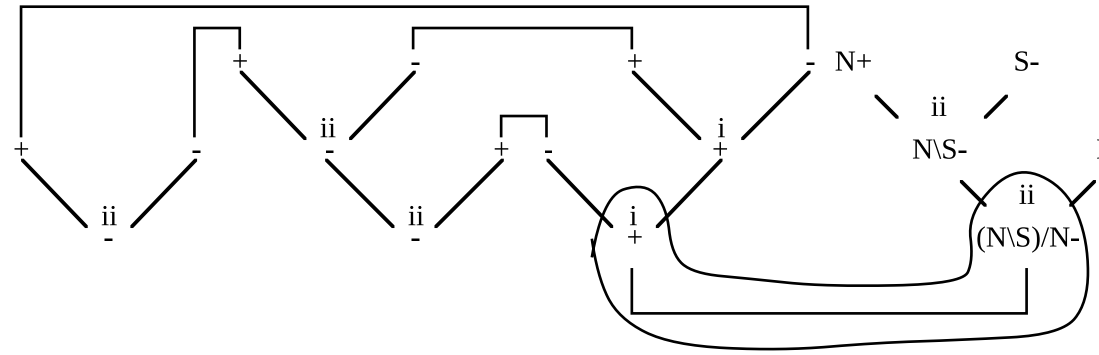
For example, the lexical semantics of ‘inhabits’can be unfolded as follows:

(48)

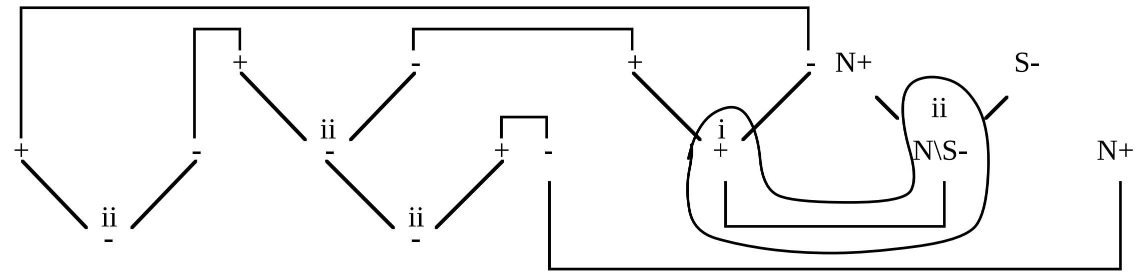


The result of such unfolding of lexical semantics can be substituted into the unfolded lexical type by a Cut, and the resulting module normalised by Cut elimination in a precompilation. Thus, for the ‘inhabits’ example:

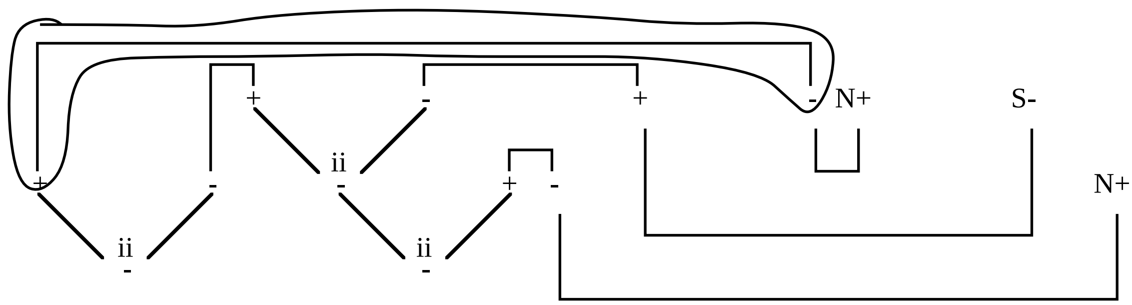
(49) a.



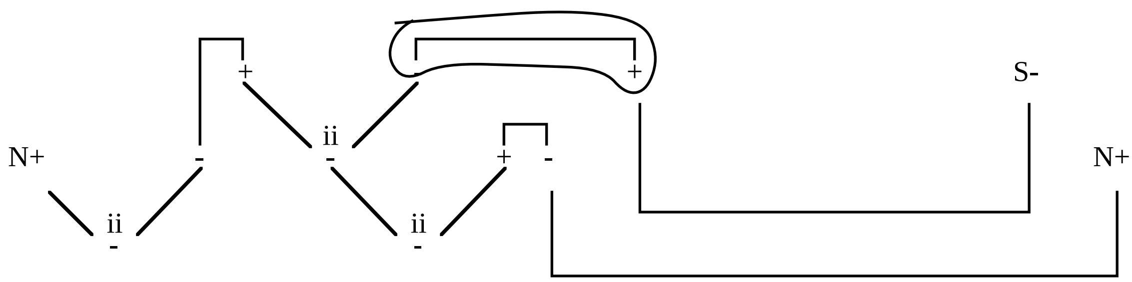
b.



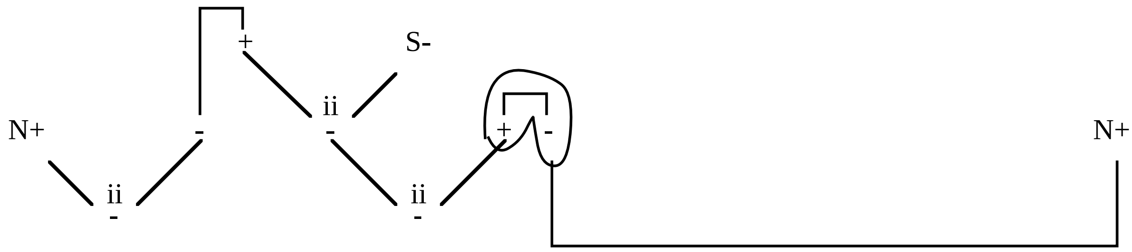
c.



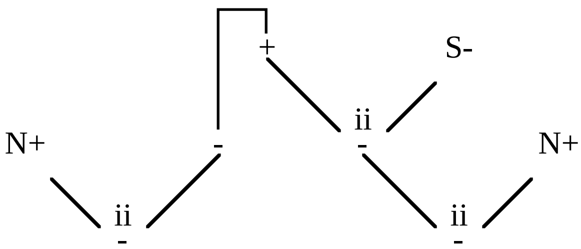
d.



e.

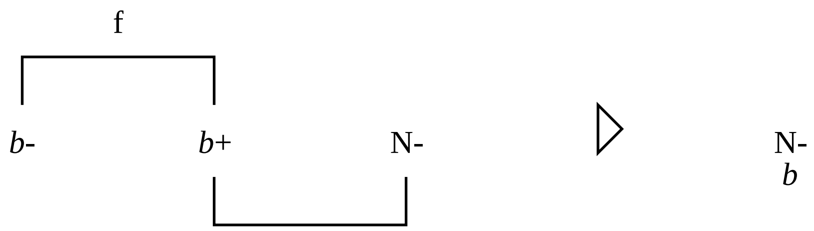


f.



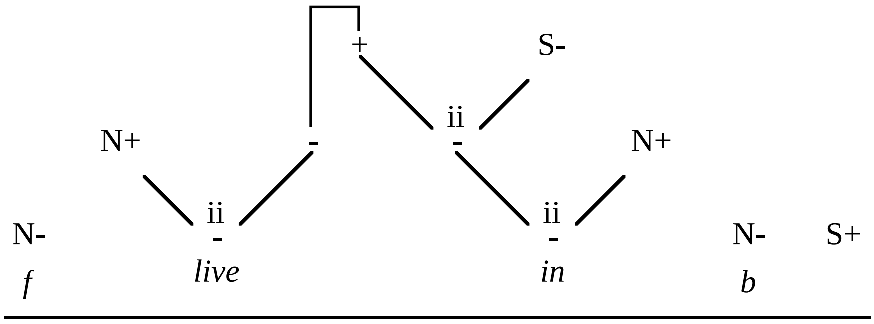
In this way, rather than starting the proof search with a frame comprising just the unfolding of lexical types, one starts with a frame comprising the pre-evaluated modules resulting from lexical substitution. Let us consider again (1b) from this point of view. First note, as well as (49), the precompilation of a proper name lexical assignment:

(50)



Thus the proof frame prior to proof search is (51a). Adding axiom links yields the net (51b) from which the semantics extracted is already normalised and is the same as that obtained from (45a) for (1a); indeed it is the same net.

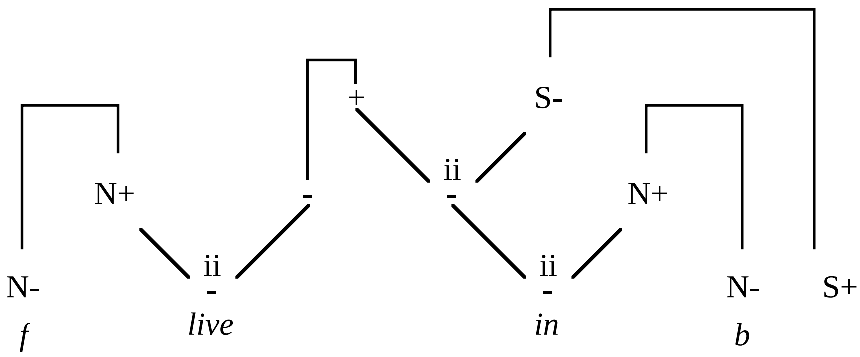
(51) a.



N, (N\S)/N, N ⇒ S

frodo+**inhabits**+**bag**+**end**: S

b.



A slightly more involved illustration of the same point is provided by the following lexical assignments for the paraphrases (3a) and (3b).

(52) a. **john** - *j*
:= N

b. **tries** - *try*
:= (N\S)/(N\S)

c. **to** - λ*xx*
:= (N\S)/(N\S)

d. **find** - *find*
:= (N\S)/N

e. **mary** - *m*
:= N

f. **seeks** - λ*x*(*try* (*x find*))
:= (N\S)/(((N\S)/N)\(N\S))

(The Montagovian higher order assignment for ‘seeks’ is motivated by the requirement to capture the two readings of (13).) These assign semantics (3c) to both (3a) and (3b) and, as the reader may check, by partially evaluating lexical modules in a precompilation, normal form semantics is obtained directly in both cases.

In both this example and the one worked out explicitly above, we deal with words which are synonyms of continuous expressions: ‘inhabits’ = ‘lives in’ and ‘seeks’ = ‘tries to find’. This enables us to represent the evaluated lexical modules as planar. However it should be noted that in general lexical substitution involves linking syntactic modules which are ordered with lexical semantic modules which are not ordered, and Cut elimination has to be performed in a hybrid architecture which must preserve the linear precedence of syntactic literals.

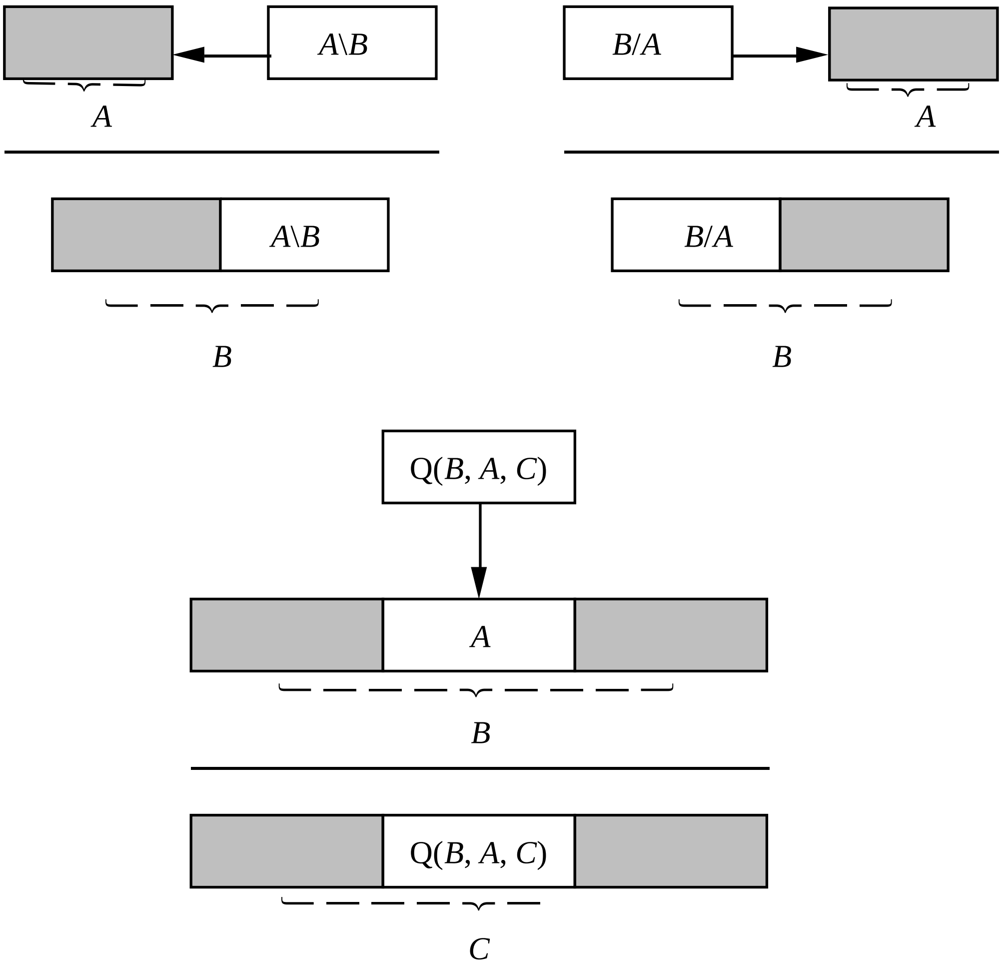
10. In situ binding (Moortgat 1990)

In the earlier sections our concerns have been for the most part programmatic, explaining and motivating the basic grammatical paradigm. In the last section a new motivation for proof nets was given. In this section we begin to address the kind of generalisation appropriate for linguistic practicality. Although the Lambek calculus **L** is not without linguistic interest (e.g. it serves for certain cases of long distance extraction, as illustrated above, as well as subcategorization) in TLG it is not the calculus itself which is central, but the logical methodology that it exemplifies. This provides a profound computational framework from which to enter into the technical task, driven by empirical requirements, of typing the constructs of linguistic generalisation. The operators $\cdot\backslash\cdot$ and $\cdot/\cdot$ are binary type constructors of *concatenation*. However, natural language also exercises discontinuous dependencies motivating type constructors for discontinuity. An effective proposal, originating in Moortgat (1990), is to add a ternary in situ binder type constructor $\mathrm{Q}(\cdot\,,\cdot,\cdot)$ of *interpolation*. Our definition (15a) of compound types is now updated to the following:

(53) $F ::= A \mid F\backslash F \mid F/F \mid F\bullet F \mid \mathrm{Q}(F, F, F)$

We can represent the action of the operators intuitively as follows:

(54)



Expressions of type *A**B* suffix themselves to expressions of type *A* thereby forming a *B*; expressions of type *B*/*A* prefix themselves to expressions of type *A* thereby forming a *B*. Now, expressions of type Q(*B*, *A*, *C*) interpolate themselves in expressions of type *B* containing an *A*, substituting themselves for the *A* and thereby forming a *C*; i.e. $[[\mathrm{Q}(B, A, C)]] = \{s \mid \forall\langle s_1, s_2\rangle \in L^2, (\forall s' \in [[A]] s_1+s'+s_2 \in [[B]]) \Rightarrow s_1+s+s_2 \in [[C]]\}$. Semantically Q(*B*, *A*, *C*) is of type $(\tau_A \rightarrow \tau_B) \rightarrow \tau_C$, applying to the abstraction of *B* over *A*. For example, quantifer phrases such as ‘someone’ and ‘every man’ will be of type Q(S, N, S) where *B* and *C* happen to be equal: they substitute themselves at the positions of names in sentences. A quantifier like ‘every’ will have type Q(S, N, S)/CN: forming a quantifier phrase after combining with a common noun to the right.

As observed by Herman Hendriks (p.c.) a sequent such as (55) is surely valid, since something which interpolates in an S can interpolate in N\\S, an incomplete S, to yield a result which is incomplete in the same sense.

(55) $\mathrm{Q}(\mathrm{S}, \mathrm{N}, \mathrm{S}) \Rightarrow \mathrm{Q}(\mathrm{N}\backslash\mathrm{S}, \mathrm{N}, \mathrm{N}\backslash\mathrm{S})$

(Where a is the semantics of the antecedent, the succedent has semantics $\lambda y \lambda z(x\ \lambda w((y\ w)\ z))$.) However, there appears to be no sequent calculus for **L**+{Q} that is satisfactory in this respect; (56), for example, does not generate (55).

(56)

$$\frac{\Gamma \Rightarrow A \qquad B \Rightarrow C}{\Gamma \Rightarrow Q(B, A, C)}\,QR \qquad\qquad \frac{\Gamma1, A, \Gamma2 \Rightarrow B \qquad \Delta1, C, \Delta2 \Rightarrow D}{\Delta1, \Gamma1, Q(B, A, C), \Gamma2, \Delta2 \Rightarrow D}\,QL$$

Our analysis of this problem is that the operator Q really combines two inference steps, an extraction and an infixation, and that incompleteness results from trying to treat Q as a unit in sequent calculus because that means proofs can only be constructed which perform these two inferences in immediate succession: not all validities have such proofs. The problem can be resolved by treating Q as a defined connective in more general systems (for example, in a sorted associative discontinuity calculus, Morrill and Merenciano 1996, or a non-associative discontinuity calculus with modalities, Moortgat 1996). But since proof nets are already motivated by the kinds of computational considerations made earlier, it is interesting to ask whether it might be possible to formulate proof nets for Q as a unit irrespective of possible decompositions and the question of proof nets for more general systems.

We have already observed that the ordering of nets for **L** corresponds to the non-commutativity of concatenation. For partial commutativity and discontinuity in general, or the Q in situ binder in particular, we cannot expect to preserve exactly the same form of ordering and planarity. One option already pursued (Moortgat 1992, Oehrle 1995, Morrill 1995) is that of dropping all ordering and regulating word order by *labelling* (Gabbay 1996) **LP** proof nets with terms subject to unification checks. However, this increases enormously the size of the search space for linking. We do not really want to abandon ordering altogether only to have to somehow regain it.

Our central suggestion is based on the following observations. In **L** the horizontal arrangement of i- and ii-cells defines a total order < on the literals of a sequent and axiom linking must be planar in this ordering. An axiom linking can be looked at as an involutive operation f on the literal occurrences: every literal is linked to exactly one other, which is in turn linked to itself. The planarity requirement is that $\sim\exists x, y,\ x<y<f(x)<f(y)$. In **LP** we can consider the literals to be totally unordered, *but still subject to the planarity requirement*; it is just that the literals being unordered, no linking will be prohibited. This provides us with a unified outlook on partial commutativity with commutativity and non-commutativity as limit cases:

(57) a. i. In the non-commutative calculus **L** the literals are totally ordered by cells.

ii. In the commutative calculus **LP** the literals are totally unordered by cells.

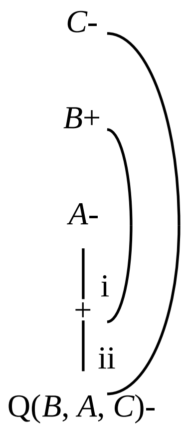
iii. In a partially commutative calculus literals will be *partially ordered* by cells.

b. The planarity requirement $\sim\exists x,\ y,\ x<y<f(x)<f(y)$, that axiom linking be planar in the partial order < determined by cells, applies in all cases.

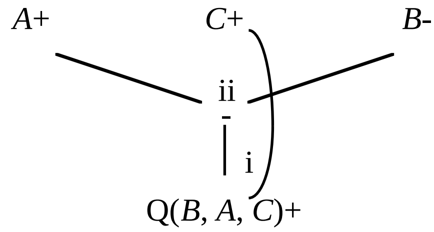
In general we shall want to define the partial order $|\cdot|$ on polar literal occurrences defined by a polar type, and hence a sequent: $|A_1,\ \ldots,\ A_n \Rightarrow A| = |A_n\backslash(\ldots(A_1\backslash A)\ldots)+|$. We have $|Ap| = \langle\{Ap\}, \varnothing\rangle$ where A is a literal occurrence of polarity p. Then where l_{Ap} and $<_{Ap}$ are the literal occurrences and their partial ordering of type A of polarity p, we have e.g. $|A{-}\!\circ B+| = \langle l_{A-}\cup l_{B+}, <_{A-}\cup<_{B+}\rangle$ but $|A\backslash B+| = \langle l_{A-}\cup l_{B+}, <_{A-}\cup<_{B+}\cup(l_{A-}\times l_{B+})\rangle$.

We propose in particular the following unfolding for Q:

(58) a.



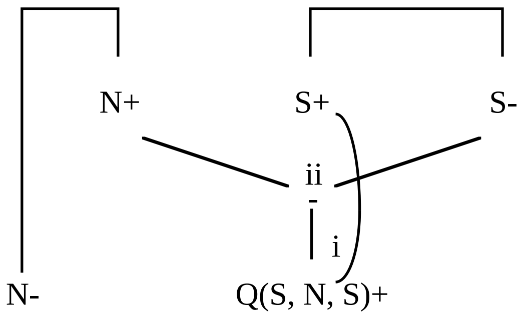
b.



The negative type is unfolded into its three component parts which are stacked one above the other, indicating that they are not ordered with respect to one another. The positive type is unfolded horizontally. That is to say $|\mathrm{Q}(B,\ A,\ C)-| = \langle l_{A-}\cup l_{B+}\cup l_{C-},\ <_{A-}\cup<_{B+}\cup<_{C-}\rangle$ and $|\mathrm{Q}(B,\ A,\ C)-| = \langle l_{A+}\cup l_{B-}\cup l_{C+},\ <_{A+}\cup<_{B-}\cup<_{C+}\cup(l_{A+}\times l_{C+})_+\cup(l_{A+}\times l_{B-})_+\cup(l_{C+}\times l_{B-})\rangle$. Note that there is a node unlabelled by any syntactic type; this corresponds to an implicit, discontinuous element, the context (shaded grey) in (54). The vertical unfolding means that the dependencies entered into by one subtype do not constrain those entered into by another. The straight and curved edges enter into (non-)circularity and the extraction of semantics just as before.

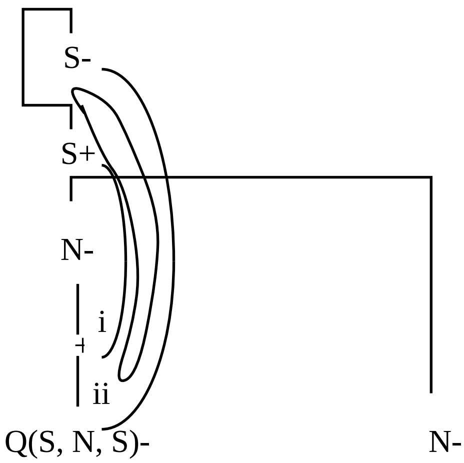
As a first example, there is the following proof net for the Q-lifting $\mathrm{N} \Rightarrow \mathrm{Q(S, N, S)}$:

(59)



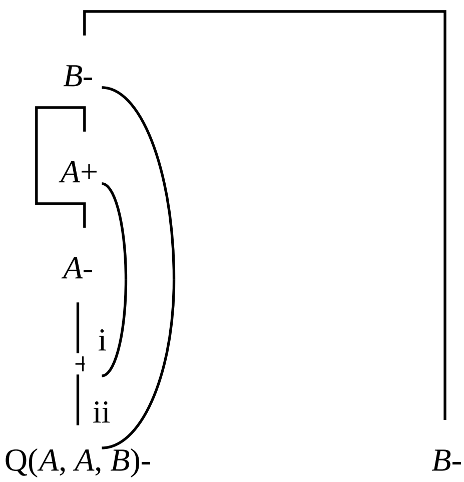
The semantics in terms of that a of the antecedent is $\lambda x(x\ \ a)$ as expected and required. Note that the Q-lowering Q(S, N, S) $\Rightarrow$ N is invalid and accordingly the following proof structure contains the circularity indicated.

(60)



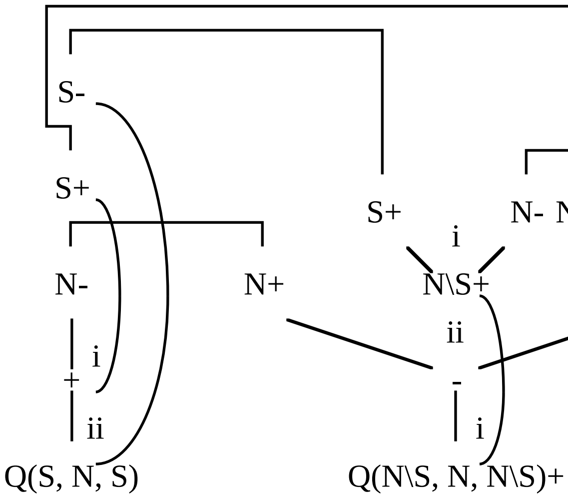
As a second example we prove in (61) a: Q(A, A, B) $\Rightarrow$ (a λxx): B.

(61)



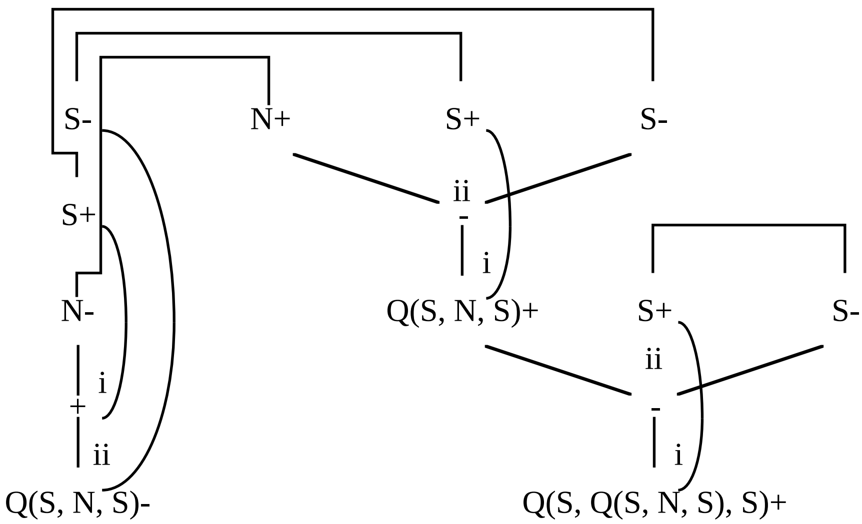
As a third example we give a proof net for the theorem (56) which could not be derived in sequent calculus:

(62)

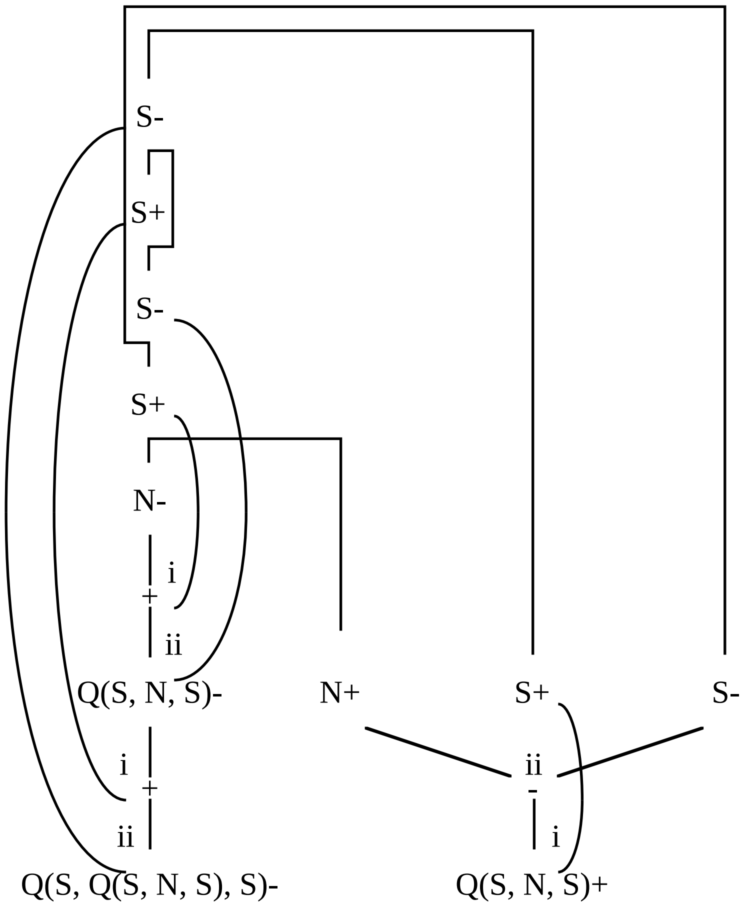


This proof net has semantics $\lambda y \lambda z(a\ \lambda w((y\ w)\ z))$ as required. Finally by way of general properties, we give the proofnets showing that Q-lifting is a closure operation: Q(S, N, S) ⇔ Q(S, Q(S, N, S), S):

(63) a.



b.

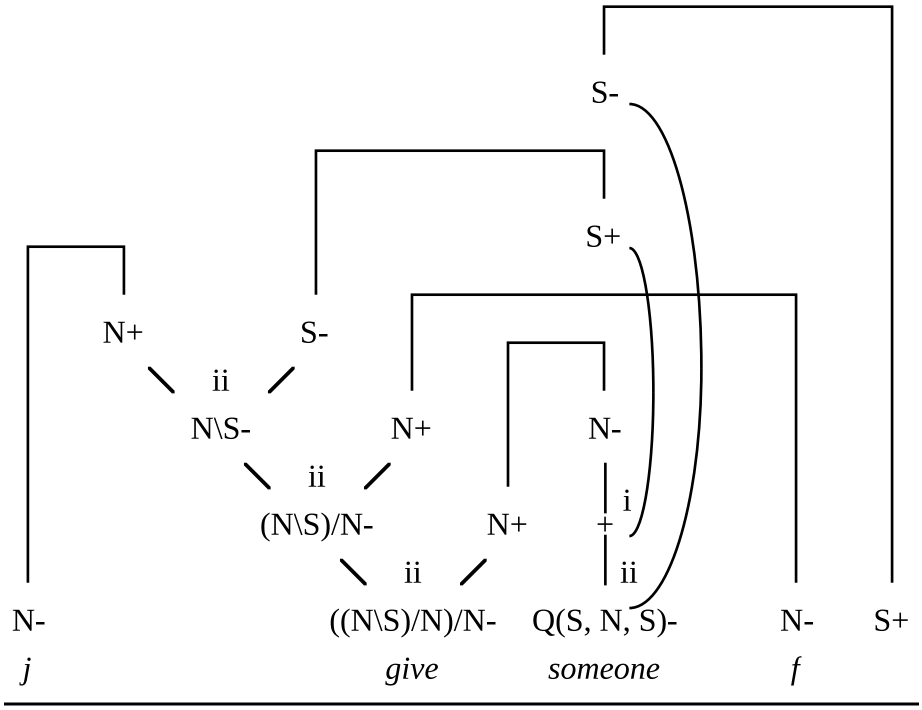


Let us consider the linguistic example of quantification; lexical type assignments are as follows.

(64)	a.	**everyone**	-	*everyone*
			:=	Q(S, N, S)
	b.	**someone**	-	*someone*
			:=	Q(S, N, S)

Example (8a) has the analysis (65). Note how the quantifier phrase unfolding is transparent to the verb's communication with its remote object.

(65)



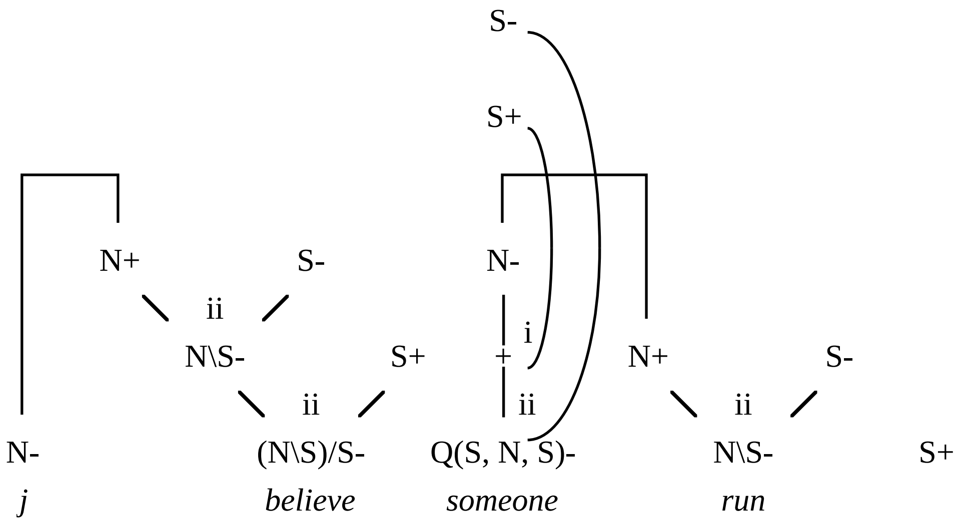
N, ((N\S)/N)/N, Q(S, N, S), N ⇒ S

john+**gave**+**someone**+**fido**: S

As the reader may check, travel according to the semantic extraction algorithm yields the expected semantic form (8b).

For the example (9) of *de re*/*de dicto* quantificational ambiguity the proof frame is (66).

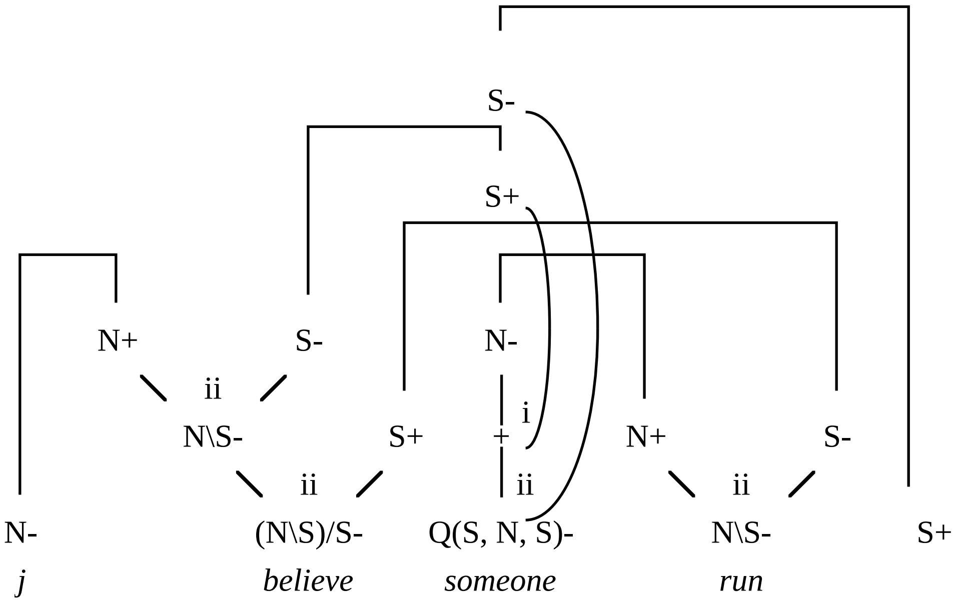
(66)



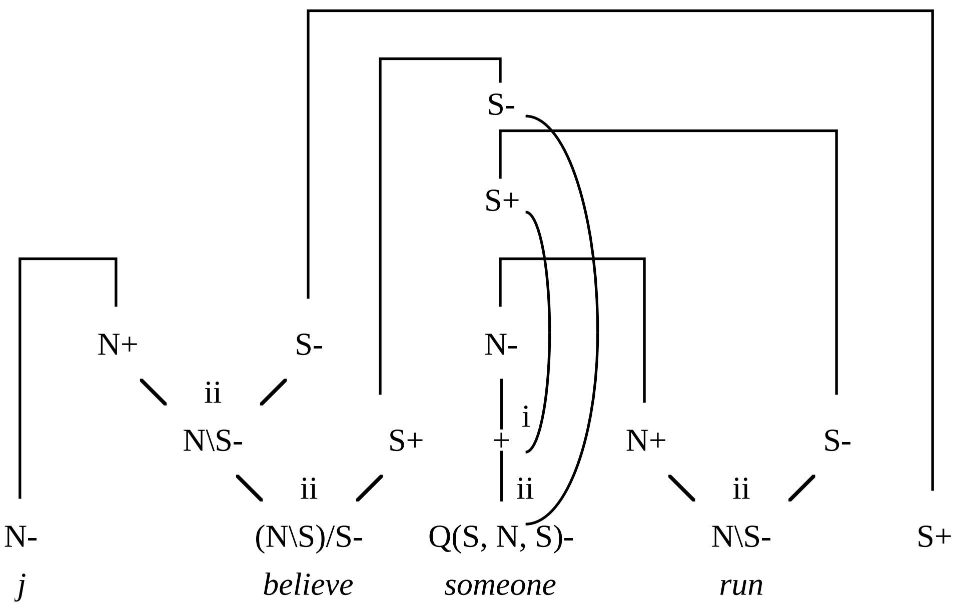
$$\frac{\text{N, (N\textbackslash S)/S, Q(S, N, S), N\textbackslash S} \Rightarrow \text{S}}{\textbf{john}+\textbf{believes}+\textbf{someone}+\textbf{runs}\text{: S}}$$

From this we can construct the *de re* analysis (67) and the *de dicto* analysis (68).

(67)



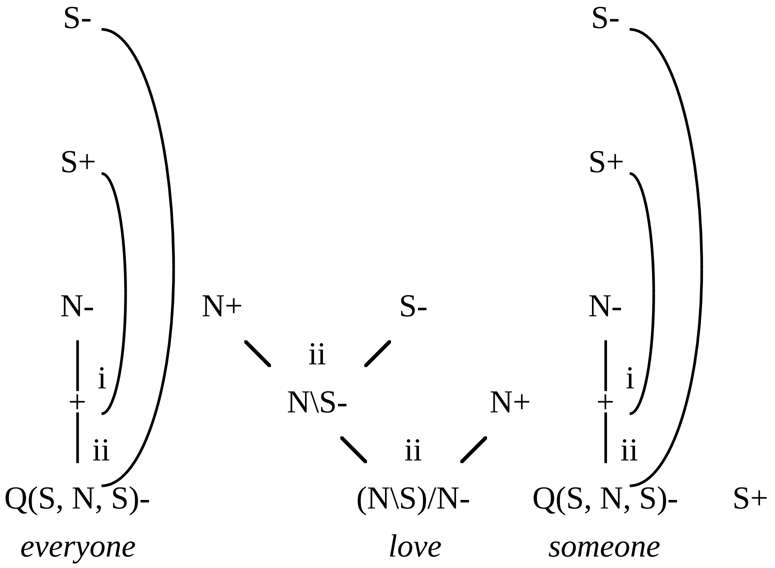
(68)



These have semantics (10) and (11) respectively.

For the example (14) of subject versus object wide scope quantificational ambiguity the proof frame is (69).

(69)

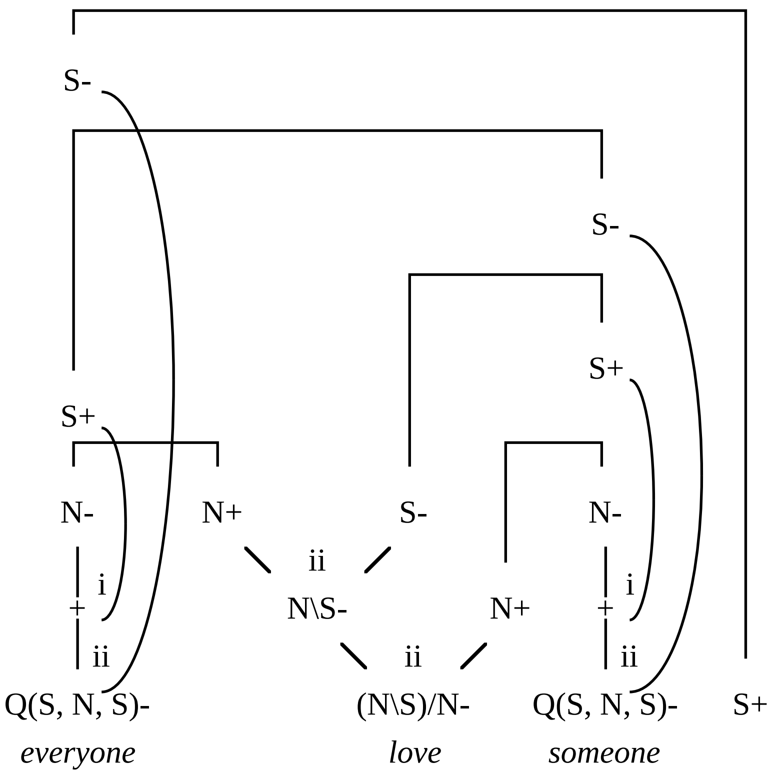


Q(S, N, S), (N\S)/N, Q(S, N, S) $\Rightarrow$ S

everyone+**loves**+**someone**: S

From this, the subject wide scope analysis is constructed in (70).

(70)



We leave to the reader the object wide scope analysis, and checking of the semantics, and consideration of (12) and (13) according to (52).

A second kind of illustration is provided by the application of Q to pied piping (Morrill 1994). We have already seen how the assignment (71a) generates (7a).

(71) a. **that** - *rel*

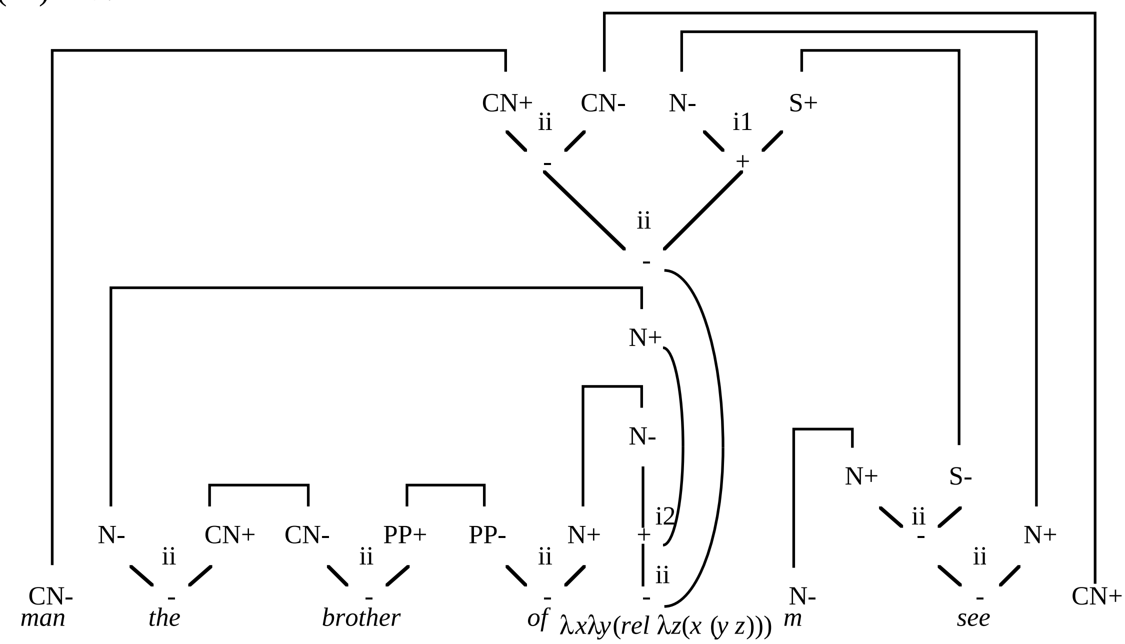
:= (CN\CN)/(S/N)

b. **whom**- $\lambda y \lambda x(rel\ \lambda z(x\ (y\ z)))$

:= Q(N, N, (CN\CN)/(S/N))

We shall see now that the single assignment (71b) is adequate for both pied piping as in (5) and non-pied piping as in (4a). The former is as follows:

(72) a.



CN, N/CN, CN/PP, PP/N, Q(N, N, (CN\CN)/(S/N)), N, (N\S)/N ⇒ CN

man+**the**+**brother**+**of**+**whom**+**mary**+**saw**: CN

b. $(((\lambda y \lambda x(rel\ \lambda z(x\ (y\ z)))\ \lambda x_2(the\ (brother\ (of\ x_2))))\ \lambda x_2((see\ x_1)\ m))\ man) = ((rel\ \lambda z((see\ (the\ (brother\ (of\ z))))\ m))\ man)$

That (71b) is also adequate for non-pied piping can be seen from the fact that an assignment of the form (71a) can be derived from (71b); that is in fact already proved in

(61), and accordingly applying the lexical semantics of (71b) to the identity function we obtain the lexical semantics of (71a):

(73) $(\lambda y \lambda x(rel\ \lambda z(x\ (y\ z)))\ \lambda x_1 x_1) =$
$\lambda x(rel\ \lambda z(x\ z)) =$
$\lambda x(rel\ x) =$
rel

The converse on the other hand is not the case: no proof net can be constructed for the frame corresponding to (7b).

References

van Benthem, J. (1991/95), *Language in Action. Categories, Lambdas, and Dynamic Logic*, North-Holland Amsterdam/MIT Press, Cambridge, Massachusetts.

Bresnan, Joan W. (1982), *The Mental Representation of Grammatical Relations*, MIT Press, Cambridge, Massachusetts.

Carpenter, Bob (1997) *Type-Logical Semantics*, to appear, MIT Press, Cambridge, Massachusetts.

Chomsky, Noam (1995) *The Minimalist Program*, MIT Press, Cambridge, Massachusetts.

Gabbay, D. (1996), *Labelled Deductive Systems*, Oxford University Press, Oxford.

Gazdar, Gerald, Ewan Klein, Geoffrey Pullum, and Ivan Sag (1985), *Generalised Phrase Structure Grammar*, Basil Blackwell, Oxford.

Girard, Jean-Yves (1987), 'Linear Logic', *Theoretical Computer Science* **50**, 1--102.

de Groote, Philippe and Christian Retoré, 'On the Semantic Readings of Proof-Nets', in G.J. Kruijff, G. Morrill and D. Oehrle (eds.), *Proceedings of Formal Grammar 1996*, Prague, 57--70.

König, E. (1989), 'Parsing as natural deduction', *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, Vancouver.

Hendriks, Herman (1993), *Studied Flexibility: Categories and Types in Syntax and Semantics*, Ph.D. thesis, Universiteit van Amsterdam.

Hepple, Mark (1990), 'Normal form theorem proving for the Lambek calculus', *Proceedings of COLING 1990*, Stockholm.

Lamarche, F. (1995), 'Games semantics for full propositional linear logic', *in Ninth Annual IEEE Symposium on Logic in Computer Science*, IEEE Press.

Lambek, J. (1958), 'The mathematics of sentence structure', *American Mathematical Monthly* **65**, 154--170; also in Buszkowski, W., W. Marciszewski, and J. van

Benthem (1988, eds.), *Categorial Grammar*, Linguistic & Literary Studies in Eastern Europe Volume 25, John Benjamins, Amsterdam, 153--172.

Lecomte, Alain and Christian Retoré (1995), ‘Pomset logic as an alternative categorial grammar’, in G. Morrill and D. Oehrle (eds.), *Proceedings of Formal Grammar 1995*, Barcelona, 181--196.

Moortgat, Michael (1988), *Categorial Investigations: Logical and Linguistic Aspects of the Lambek Calculus*, Foris, Dordrecht.

Moortgat, Michael (1990), ‘The Quantification Calculus: Questions of Axiomatisation’, Deliverable R1.2.A, DYANA Dynamic Interpretation of Natural Language, ESPRIT Basic Research Action BR3175.

Moortgat, Michael (1991/96), ‘Generalised Quantification and Discontinuous Type Constructors’, in Bunt and van Horck (eds.) *Discontinuous Constituency*, Mouton de Gruyter, Berlin, 181--208.

Moortgat, Michael (1992), ‘Labelled Deductive Systems for categorial theorem proving’, *Proceedings of the Eighth Amsterdam Colloquium*, Amsterdam, 403--424.

Moortgat, Michael (1996), ‘In situ binding: A modal analysis’, in P. Dekker and M. Stockhof (eds.*) Proceedings of the Tenth Amsterdam Colloquium*, Amsterdam, 539--549.

Moortgat, Michael (1997), ‘Categorial type logics’, in J. van Benthem and A. ter Meulen (eds.) *Handbook of Logic and Language*, Elsevier, Amsterdam, 93--177.

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

Morrill, G. (1995), ‘Clausal proofs and discontinuity’, *Bulletin of the Interest Group in Pure and Applied Logic* **3**, *2*, *3*, 403--427.

Morrill, G. (1996), ‘Memoisation of categorial proof nets: parallelism in categorial processing’, research report LSI--96--24--R, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona.

Morrill, G. and J.M. Merenciano (1996) ‘Generalising discontinuity’, *Traitement automatique des langues* **37**, *2*, 119--143.

Oehrle, R. T. (1995), ‘Term-labeled categorial type systems’, *Linguistics and Philosophy* **17**, 633--678.

Pentus, M. (1993), ‘Lambek calculus is L-complete’, ILLC report, Universiteit van Amsterdam

Prawitz, D. (1965), *Natural Deduction: A Proof-Theoretical Study*, Almqvist and Wiksell, Uppsala.

Roorda, Dirk (1991), *Resource Logics: Proof-theoretical Investigations*. Ph.D. thesis, Universiteit van Amsterdam.

Steedman, Mark (1985), ‘Dependency and Coordinacion in the Grammar of Dutch and English’, *Language* **61**, 523--568.