Geometry of Lexico-Syntactic Interaction

Glyn Morrill
Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Jordi Girona Salgado, 1-3
E-08034, Barcelona
morrill@lsi.upc.es

Abstract

Interaction of lexical and derivational semantics ---for example substitution and lambda conversion--- is typically a part of the on-line interpretation process. Proof-nets are to categorial grammar what phrase markers are to phrase structure grammar: unique graphical structures underlying equivalence classes of sequential syntactic derivations; but the role of proof-nets is deeper since they integrate also semantics. In this paper we show how interaction of lexical and derivational semantics at the lexico-syntactic interface can be precomputed as a process of off-line lexical compilation comprising Cut elimination in partial proof-nets.

Introduction

Consider the following examples of paraphrase:

(1) a. Frodo lives in Bag End.
b. Frodo inhabits Bag End.
c. ((*in b*) (*live f*))

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

Typically, for at least (1b) and (2b) the normalised semantic forms result from a process of substitution and lambda conversion subsequent to or simultaneous with syntactic derivation. We show how such interaction of lexical and derivational semantics at the lexico-syntactic interface can be precomputed as a process of off-line lexical compilation comprising Cut elimination in partial proof-nets.

For accessibility, we devote in the initial sections a considerable proportion of space to an introduction to categorial grammar oriented towards proof-nets; see also Morrill (1994), Moortgat (1996) and Carpenter (1997).

1 Categorial grammar

We consider categorial grammar with category formulas $\mathbb{F}$ (categories) defined by the following grammar:

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

The categories in $\mathbb{A}$ are referred to as atomic 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 S and names N; what the class is exactly is not fixed by the formalism.

Left division categories $A\backslash B$ ('A under B') are those of expressions (functors) which concatenate with (arguments) in A on the left to yield Bs. Right division categories B/A ('B over A') are those of expressions (functors) which concatenate with (arguments) in A on the right yielding Bs. Product categories $A\bullet B$ are those 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 category formula A is interpreted as a subset $[[A]]$ of L. When the interpretation of atomic categories has been fixed, that of complex categories is defined by (4).

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

[1] In fact Lambek (1958) excluded the empty string ---and hence empty antecedents in the calculus of (5)--- but it is convenient to include it here.

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

2 Lambek sequent calculus

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

(5) a.

$$\frac{}{A \Rightarrow A}\ \text{id}$$

$$\frac{\Gamma \Rightarrow A \qquad \Delta 1, A, \Delta 2 \Rightarrow C}{\Delta 1, \Gamma, \Delta 2 \Rightarrow C}\ \text{Cut}$$

b.

$$\frac{A, \Gamma \Rightarrow B}{\Gamma \Rightarrow A \backslash B}\ \backslash \text{R}$$

$$\frac{\Gamma \Rightarrow A \qquad \Delta 1, B, \Delta 2 \Rightarrow C}{\Delta 1, \Gamma, A \backslash B, \Delta 2 \Rightarrow C}\ \backslash \text{L}$$

c.

$$\frac{\Gamma, A \Rightarrow B}{\Gamma \Rightarrow B / A}\ / \text{R}$$

$$\frac{\Gamma \Rightarrow A \qquad \Delta 1, B, \Delta 2 \Rightarrow C}{\Delta 1, B/A, \Gamma, \Delta 2 \Rightarrow C}\ / \text{L}$$

d.

$$\frac{\Gamma 1 \Rightarrow A \qquad \Gamma 2 \Rightarrow B}{\Gamma 1, \Gamma 2 \Rightarrow A \bullet B}\ \bullet \text{R}$$

$$\frac{\Gamma 1, A, B, \Gamma 2 \Rightarrow C}{\Gamma 1, A \bullet B, \Gamma 2 \Rightarrow C}\ \bullet \text{L}$$

$\Gamma(n)$ and $\Delta(n)$ range over *context* sequences of category formulas; A, B, and $A*B$ are referred to as the *active* formulas. The calculus **L** lacks the usual structural rules of permutation, contraction and weakening. Adding permutation collapses the two divisions into a single non-directional implication 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, processing can be performed using just the left (L) and right (R) rules. These rules all decompose active formulas $A*B$ in the left or the right of the conclusions into subformulas A and B in the premises, and 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:

(6)

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

Where a labels the antecedent, the coding of this proof as a lambda term ---what we

[2] The completeness of the calculus with respect to the intended interpretation was proved in Pentus (1994).

[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).

shall call the derivational semantics--- is $\lambda x(x\ a)$. The converse of lifting, lowering, in (7) is not derivable. A proof of a theorem of composition (it has as its semantics functional composition) is given in (8).

(7) $S/(N\backslash S) \Rightarrow N$

(8)

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

A grammar contains a set of lexical assignments α: A. An expression $w_1+\ldots+w_m$ is of category 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 and transitive verbs, there are the following derivations:

(9)

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

(10)

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

Ungrammaticality occurs when there is no validity of the sequents arising by lexical insertion, as in the following:

(11)

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

3 Ambiguity and spurious ambiguity

The sentence (12) is structurally ambiguous.

(12) 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:

(13) a.

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

b.

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

c.

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

However, sometimes a non-ambiguous expression also has more than one sequent proof (even excluding Cut); thus the sequent in (14a) has the proofs (14b) and (14c).

(14) a.

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

b.

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

c.

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

As the reader may check, $N/CN, CN \Rightarrow S/(N\backslash S)$ 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 sequentialisation of inferences ---in the case of (14) /L and \L--- when in fact they are not ordered by dependency and can be performed in parallel.

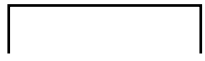
The problem can be resolved by defining stricter normalised proofs which impose a unique ordering when alternatives would otherwise be available (König 1990, Hepple 1990, Hendriks 1993). However, while this removes spurious ambiguity as a problem arising from independence of inferences, it signally fails 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 sequentialisation, but as an opportunity for parallelism. This opportunity is grasped in *proof-nets*.

4 Lambek proof-nets

Proof-nets for **L** were developed by Roorda (1991), adapting their original introduction for linear logic in Girard (1987). In proof-nets, the opposition of formulas arising from their location in either the antecedent or the succedent of sequents is replaced by assignment of polarity: input (negative) for antecedent and output (positive) for succedent. A proof-net is a kind of graph of polar formulas.

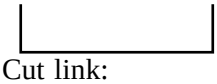
First we define a more general concept of *proof structure*. These are graphs assembled out of the following *links*:

(15) a.



X $\sim X$

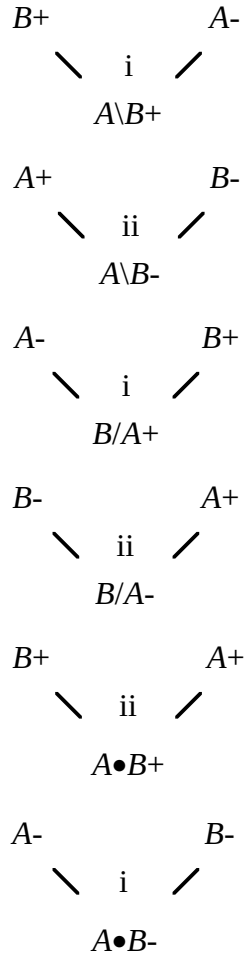
id link:
zero premises,
two conclusions



X $\sim X$

Cut link:
two premises,
zero conclusions

b.



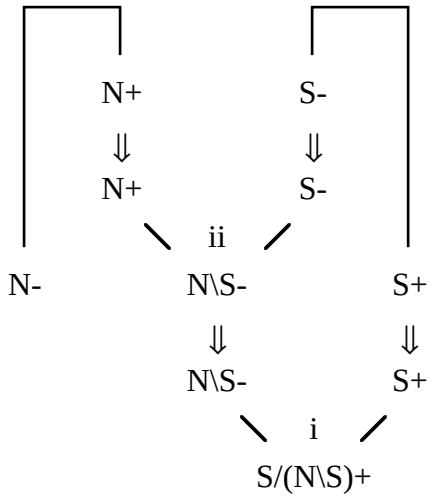
i- and ii-links:
two premises,
one conclusion

In the id and Cut links X and $\sim X$ schematise over occurrences of the same category with opposite polarity. Note that the nodes of links are also marked (implicitly) as being either conclusions (looking down) or premises (looking up). In the i- and ii-links the middle nodes are the conclusions and the outer nodes the premises. The i-links correspond to unary sequent rules and the ii-links to binary sequent rules. Observe that in the output, but not in the input, unfoldings the order of subformulas is switched between premises and conclusion; this is essential to the characterization of ordering by graph planarity.

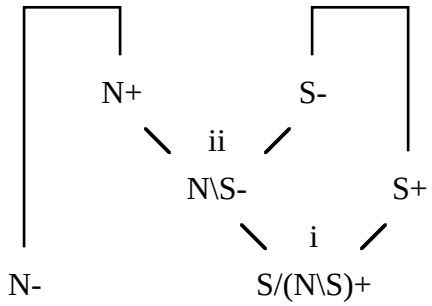
Proof structures are assembled by identifying nodes of the same polar category which are the premises and conclusions of different components; premises and conclusions not fused in this way are the premises and conclusions of

the proof structure as a whole. For example, in (16a) four links are assembled into a proof structure (16b) with no premises and two conclusions, N- and S/(N\S)+:

(16) a.



b.



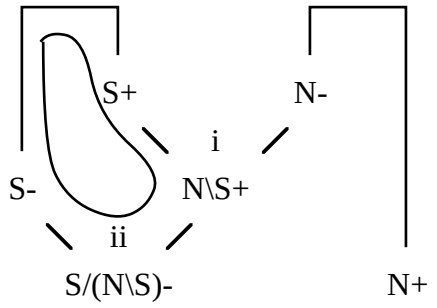
Proof-nets are proof structures which arise, essentially, by forgetting the contexts of the sequent rules and keeping only the active formulas, but not all proof structures are well-formed as proofs. There must exist a global synchronization of the partitioning of contexts by rules (the long trip condition of Girard 1987). Eschewing the (somewhat involved) details (Danos and Regnier 1990; Bellin and Scott 1994) it suffices here to state that a proof structure is well-formed, a *module* (partial proof-net), iff every cycle crosses both edges of some i-link. A module is a *proof-net* iff it contains no premises. The structure (16b) is a proof-net, in fact it is the proof-net for our instance (6) of lifting since its conclusions are the polar categories for this sequent:

(17)

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

The structure in (18) is not a module because it contains the circularity indicated: it corresponds to the lowering (7), which is invalid.

(18)



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

The structure of figure 1 is a module with two premises and three conclusions; the latter are the polar categories of our composition theorem (8). Adding the remaining id axiom link makes it a proof-net for composition.

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.

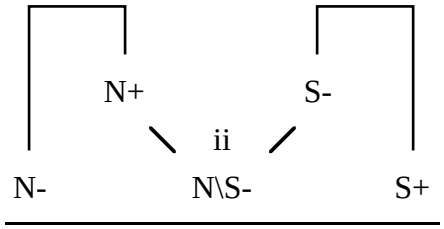
Like the sequent calculus, proof-nets enjoy the Cut elimination property whereby every proof has a Cut-free equivalent. The evaluation of a net to its Cut-free normal form is a process of graph reduction. The reductions are as shown in figure 2.

5 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.

The analysis of an expression as search for βη-long proof-nets can be construed in three phases, 1) selection of lexical categories for elements in the expression, 2) unfolding of these categories into a *frame* of trees of i- and ii-links with atomic leaves (literals), and 3) addition of (planar) id axiom links to form proof-nets. For example, ‘John walks’ has the following analysis:

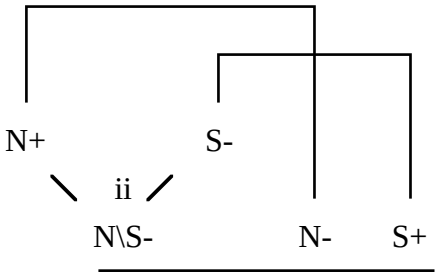
(19)



$$\frac{N, N\backslash S \Rightarrow S}{\textbf{john}+\textbf{walks}:S}$$

The ungrammaticality of 'walks John' is attested by the non-planarity of the proof structure (20).

(20)



$$\frac{N\backslash S, N \Rightarrow S}{\textbf{walks}+\textbf{john}:S}$$

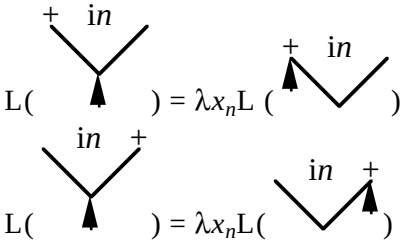
As expected, where there is structural ambiguity there are multiple derivations; see figure 3. But now also, when there is no structural ambiguity there is only one derivation, as in figure 4. This property is entirely general: the problem of spurious ambiguity is resolved.

6 Proof-net semantic extraction

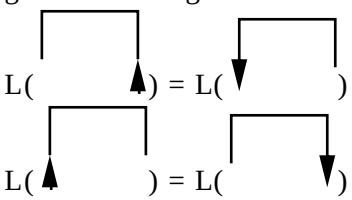
Until now we have not been explicit about how a proof determines a semantic reading. We shall show here how to extract from a proof-net a functional term representing the semantics (see de Groote and Retoré 1996, who reference Lamarche 1995). This is done by travelling through a proof-net and constructing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

First one assigns a distinct variable index to each i-link; then one starts travelling upwards through the unique positive conclusion. Thereafter the function *L* mapping proof-nets to lambda terms is as follows (for brevity we exclude product):

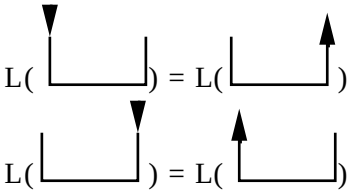
(21) a.
Going up through the conclusion of a i-link, make a functional abstraction for the corresponding variable and continue upwards through the positive premise:



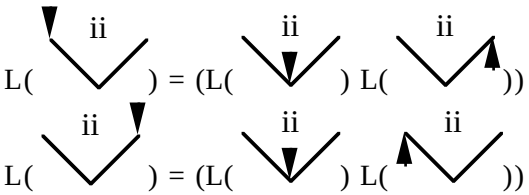
b.
Going up through one id conclusion, go down through the other:



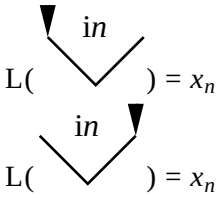
c.
Going down through one premise of Cut, go up through the other:



d.
Going down through one premise of a ii-link, 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-link, put the corresponding variable:



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

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

(22)

frodo	-	*f*
	:=	N
lives	-	*live*
	:=	N\S
in	-	*in*
	:=	(S\S)/N
bag+**end**	-	*b*
	:=	N
inhabits	-	λ*x*λ*y*((*in x*) (*live y*))
	:=	(N\S)/N

Then (1a) has the analysis given in figure 5, with semantic extraction (23), where * marks the point at construction and Roman numerals indicate the argument traversals, performed after the function traversals, triggered by entry into ii-links.

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

Example (1b) has the analysis given in figure 6, for which the semantic extraction is (24).

(24) (* 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 (23) 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:

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

Although such lambda conversion only calculates what the grammar defines and is not part of the grammar itself, computationally it is an on-line process. The following section shows how this can be rendered, in virtue of proof-nets, an off-line process of lexical compilation.

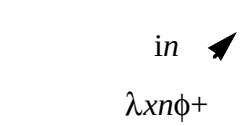
7 Off-line semantic evaluation

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 a process of 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 categories.[5]

Lexical semantics expressed as a linear (=single bind) lambda term is unfolded into an (unordered) proof-net by the algorithm (26):

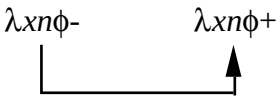
(26) a.
Start with the λ-term φ at a + node: φ+.

b.
To unfold λx_nφ+, make it the conclusion of a i-link with index *n* and unfold φ+ at the positive premise:

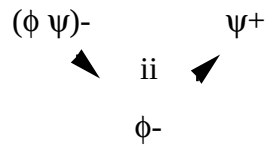


[5] Lecomte and Retoré (1995) propose to use the expressivity of modules in general to classify words rather than just category formulas (=modules without id or Cut links). Our method provides semantic motivation for modules at the machine level but we propose to maintain the less unwieldy categories at the user level.

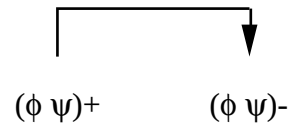
c.
To unfold $\lambda x_n \varphi$-, make it a Cut premise and unfold $\lambda x_n \varphi$+ at the other premise:



d.
To unfold $(\varphi\ \psi)$-, make it the premise of a ii-link and unfold φ- at the conclusion and ψ+ at the other premise:



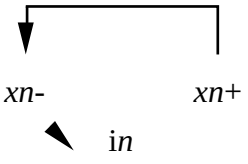
e.
To unfold $(\varphi\ \psi)$+ make it the conclusion of an id link and unfold $(\varphi\ \psi)$- at the other conclusion:



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-link with index *n*:

to unfold x_n+ make it an id premise first:



For example, the lexical semantics of 'inhabits' can be unfolded as shown in figure 7. The result of such unfolding of lexical semantics can be substituted into the unfolded lexical category by a Cut, and the resulting module normalised by Cut elimination in a precompilation. This is illustrated for the 'inhabits' example in figure 8.

In this way, rather than starting the proof search with a frame comprising just the unfolding of lexical categories, 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 figure 8, the precompilation of a proper name lexical assignment as in figure 9. The proof frame prior to proof search is that in figure 10. Adding axiom links yields the same net, and thus the same semantics, as that obtained for (1a) in figure 5.

A slightly more involved illustration of the same point is provided by the following lexical assignments for the paraphrases (2a) and (2b).

(27)

john	- *j*
	:= N
tries	- *try*
	:= (N\S)/(N\S)
to	- λ*xx*
	:= (N\S)/(N\S)
find	- *find*
	:= (N\S)/N
mary	- *m*
	:= N
seeks	- λ*x*(*try* (*x find*))
	:= (N\S)/(((N\S)/N)\(N\S))

These assign semantics (2c) to both (2a) and (2b) and, as the reader may check, by partially evaluating lexical modules in a precompilation, normal form semantics is obtained directly in both cases.

Conclusion

In both the example worked out explicitly and the one left to the reader, 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 which could be multiple-binding, and Cut elimination has to be performed in a hybrid architecture which must preserve the linear precedence of syntactic literals. It is therefore of importance to the future generalization of the method we propose to investigate the precise nature of such hybrid architectures.

Acknowledgements

My thanks to Josep Maria Merenciano for discussions relating to this work.

References

Bellin G. and Scott P. J. (1994) *On the π-Calculus and Linear Logic*. Theoretical Computer Science, 135, pp. 11--65.

Carpenter B. (1998) *Type-Logical Semantics*. MIT Press, Cambridge, Massachusetts.

Danos R. and Regnier L. (1990) *The structure of multiplicatives*. Archive for Mathematical Logic 28, pp. 181--203.

de Groote Ph. and Retoré C. (1996) *On the Semantic Readings of Proof-Nets*. In "*Proceedings of Formal Grammar 1996*", G.J. Kruijff, G. Morrill & D. Oehrle, ed., Prague, pp. 57--70.

Girard J.-Y. (1987) *Linear Logic*. Theoretical Computer Science, 50, pp. 1--102.

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

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

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

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, pp. 154--170.

Lecomte A. and Retoré C. (1995) *Pomset logic as an alternative categorial grammar*. In "Proceedings of Formal Grammar 1995", G. Morrill & D. Oehrle, ed., Barcelona, pp. 181--196.

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

Moortgat M. (1996) *Categorial type logics*. In "Handbook of Logic and Language", J. van Benthem & A. ter Meulen, ed., Elsevier, Amsterdam, pp. 93--177.

Pentus M. (1994) *Language completeness of the Lambek calculus*. Proceedings of the Eight Annual IEEE Symposium on Logic in Computer Science.

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