Proof-nets as Linguistic Structures for LFG

Avery D Andrews
ANU, Nov 2006
INCOMPLETE DRAFT (minor edits from Oct version)

Crouch and van Genabith (1999), with their 'node ordering' concept, propose to use glue derivations as a level of linguistic structure alongside of others, such as c-structure and f-structure. This idea is taken up in Asudeh and Crouch (2002), and a different relation of the same general nature, 'Outscopes' is proposed in Andrews (to appear). However the use of deductions to build semantic representations, as appears to be standard in current glue work, doesn't fit well with the rest of the architecture of LFG, with its emphasis on relatively compact and static substructures, related to each other by rich correspondence relations. Here I will show how we can produce a version of glue semantics that is more consonant with the rest of LFG architecture by using proof-nets, which can furthermore be reoriented so as to be to be very similar to conventional logical forms.

1 Proof-nets as Logical Forms

Proof-nets were originally developed as the Girard's preferred format for proofs in the multiplicative fragment of linear logic, on the basis that they eliminated various conceptually irrelevant differences between proofs (i.e. they do roughly the same thing for proofs as phrase-structure trees do for context-free grammar derivations, a feature which recommends them for use as linguistic structures).

However, the simplest kinds of proof-nets, the intuitionisitic multiplicative, and, expecially, purely implicational ones, can be arrived at on a superficially rather different basis, as an alternative format for semantic composition of meaningful elements, based on specifying input-output relationships between the (simple type-theoretical) types of the elements. For a simple illustration, consider the types of the expressions *Bert* and *Hungry* in any reasonable type-based format for semantic representations/logical forms, assuming e as the type for 'individuals/entities', and p as the type for 'propositions' (following the usage of Higher Order Grammar):

(1) *Bert* e *Hungry* $e \rightarrow p$ (e, p)

If confronted with these items, and urgently requested to do something useful with them, an obvious thing to try would be to connect the e of *Bert* as 'output' to the e of *Hungry* as 'input', like this:

(2) *Bert* e *Hungry* $e \rightarrow p$ (e, p)

But (2) is something that is already familiar in the literature,[1], a intuitionistic implicational proof-net, although it is displayed in the root-up orientation that linguists tend to favor for their trees, rather than the root-down presentation that logicians tend to use.[2] The links connecting the atomic formulas are called 'axiom-links' (since they correspond logically to the use of the 'axiom' rule in a deduction).

One might object to this representation on the basis of wanting a conventional-looking logical form rather than some sort of novel graph-structure as the outcome of semantic assembly, but this objection is vitiated by the fact that, modulo some issues concerning anaphora that we'll discuss later, multiplicative proof-nets are a perfectly good format for logical forms. This is shows by de Groote and Retoré (1996) and Perrier (1999), and can perhaps be shown more vividly by using a slight variation of the notion of 'dynamic graph' of de Groote (1999).[3]

In order to describe the formation of the dynamic graph, we will need to introduce some concepts from proof-net theory, but adjusting the terminology to be more attuned to linguistics than to the logical origins of proof-nets.[4] The version of proof-nets I will use for glue[5] is based on a collection of tree-structures determined by the semantic types of the items being combined, which I'll call a 'frame', while each tree in the frame will be called a 'type tree'.

These trees are normally described as made of 'tensor' and 'par' links (heritage from the one-sided sequent calculus), but we'll just say that their branching subtrees are 'implications' (and later, tensors), with the leaves being literals. So in (1, 2) above, one of the type-trees consists of a single literal, while the other consists of an implication with literals as left and right branches.

Next, we need a notion of 'polarity', which, unfortunately, is standardly assigned oppositely to the direction that seems natural for linguistics (at least to me); the type-trees in the frame are assigned negative polarity, and also described as 'outputs'.[6] The members of the frame are then all assigned negative polarity,[7] which is then propagated downward into the subtrees in accordance with the following principles:

(3) a. The antecedent of a negative implication is positive

b. The consequent of a negative implication is negative

c. The antecedent of a positive implication is negative

d. The consequent of a positive implication is positive

[1] Although much more widely used in Type-Logical Grammar than in LFG

[2] So that the subtending link is an axiom-link rather than a cut-link.

[3] Itself based on the 'paths' of Lamarche (1994).

[4] For which see, for example, de Groote (1999), Moot (2002), ch 4.

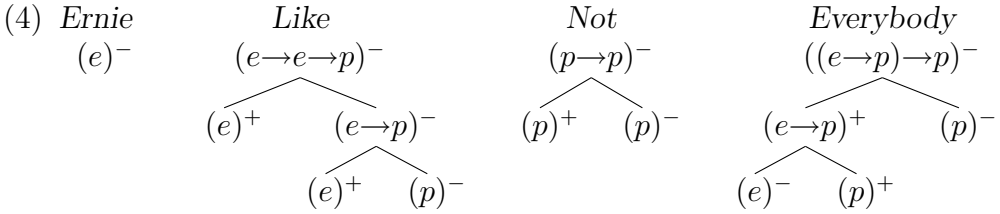
[5] There are quite a number of others available

[6] Andrews (2006, to appear) reverses the polarities, but here I'll use the standard ones in order to stay closer to the proof-net literature.

[7] In the standard formulation, there is one additional frame-member of positive polarity, which we don't really need

We can illustrate these principles with a reasonably complex example, the frame for both *Ernie doesn't like everybody* and *everybody doesn't like Ernie*:

(4)



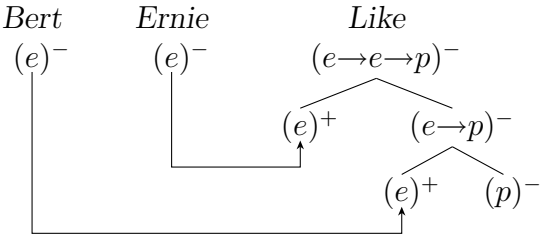
A wide range of techniques are employed in the literature for representing the connectives and their polarities.[8] The next step in building a proof-net is to match up positive and negative literals of the same type, in non-overlapping pairs, with one unmatched negative left over. This creates an intuitionistic proof-structure:

(5) A intuitionistic proof-structure is a frame in which positive and negative literals of the same type but opposite polarity are linked in non-overlapping pairs, by the addition of 'axiom links', with one negative literal left unlinked, or, alternatively, there is one and only one positive polarity formula in addition to the negative ones, and no literals are left unlinked.[9]

Since all our proof-structures will be intuitionistic (non-intuitionistic ones allow the contributing formulas to include more than one positive), we will hence omit the qualification 'intuitionistic'.

Here is a proof-structure based on a simpler frame than (4):

(6)



An issue with both this frame and (4) is that there are too many ways of hooking them up in accordance with (5). So for example, (6) has an alternative hookup that would be naturally interpreted as meaning *Ernie likes Bert*.

In LFG, this problem can be solved by supplementing the semantic type information with syntactic f-structural location information.[10] In a close-to-standard version of

[8] Most often using the tensor symbol '⊗' for the negative implications and the par symbol '⅋' for the positive ones, on the basis of how proofnets can be derived from the one-sided sequent calculus, as discussed for example in Crouch and van Genabith (2000).

[9] The first formulation is more convenient for linguistics, the second, logically cleaner.

[10] Other frameworks based on similar ideas, such as Categorial Grammar and Type-Logical Grammar make implication sensitive to linear order.

the glue meaning-constructor notation, for example, the uninstantiated semantic contributions that would lead to the hookup of (6) would be represented in the lexicon as:[11]

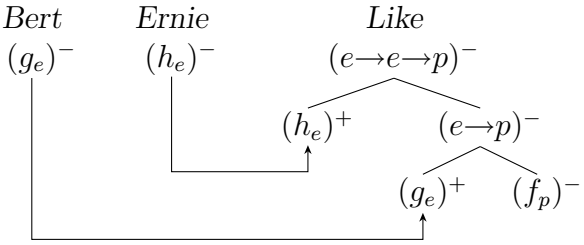
(7) $\begin{aligned} \textit{Like} &: (\uparrow \mathrm{OBJ})_e \multimap (\uparrow \mathrm{SUBJ})_e \multimap \uparrow_p \\ \textit{Bert} &: \uparrow_e \\ \textit{Ernie} &: \uparrow_e \end{aligned}$

Upon use in a syntactic structure, and consequent instantiation and resolution of functional designators, they would become:

(8) $\begin{aligned} \textit{Like} &: h_e \multimap g_e \multimap f_p \\ \textit{Bert} &: h_e \\ \textit{Ernie} &: g_e \end{aligned}$

Rendered into the type-tree format for frames in the obvious way, this becomes:

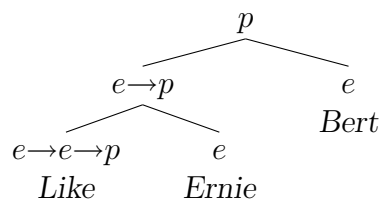
(9)



By imposing the requirement that the literals connected to make a proof-structure must agree in both semantic type and in associated f-structure information, we can enforce many of the most important constraints that syntactic structure imposes on semantic interpretation. Later we will return to the f-structure connection and say more about what it amounts to, but meanwhile we need to return to the notion of dynamic graph.

If one contemplates (9) for long enough, the thought might occur that it's sort of like a tree lying on its side, with some distortions. In particular, the unlinked literal is at the bottom-right rather than the top. And the links running upward from the right-daughters to their mothers in (9) correspond to the links that would go from the mothers to the left-daughters in more conventional expression-tree format (built in accordance with the usual rules for combining arguments of different types under the application construction) such as:

(10)

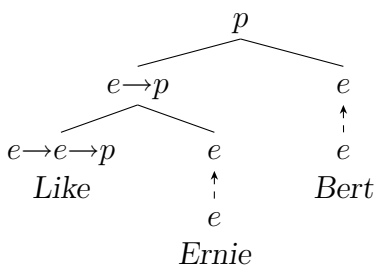


[11]Note, however, that the 'semantic projection' is being omitted, as will be discussed later.

And finally, the left daughters in the type-tree correspond to the right-daughters in the 'expression tree' (10).

The two axiom-links don't correspond to anything in the expression tree, but this can be fixed trivially, by either treating axiom-linked nodes as single nodes, or adding 'plug in' links to the expression tree, here expressed as dashed lines, indicating where arguments are plugged into positions that can be regarded as pre-prepared on the basis of the type-tree:

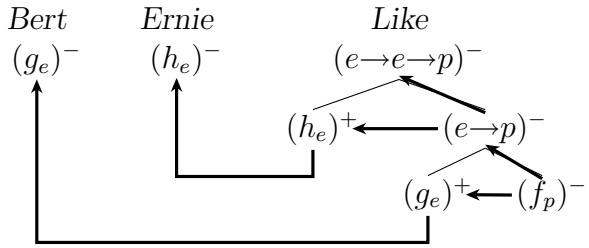
(11)



The relationship between the 'plugged' representation of (10) and the 'deplugged' one of (11) is reversible, since axiom/plug in-links only connect nodes labelled with basic types, and all such nodes except the top one.[12]

We can indicate more vividly how the expression-tree is derived from the proof-structure by drawing the new links as heavy lines, superposed on the old, making it clear that they are a re-presentation of the structure rather than an original creation:

(12)



The arrows here are those of de Groote's (1999) 'dynamic graph', except that we've reversed their direction in order to conform with linguists' intuitions about the direction of domination in trees. To indicate this difference plus one addition to come, we'll call the format of (12) an 'expression-graph'.

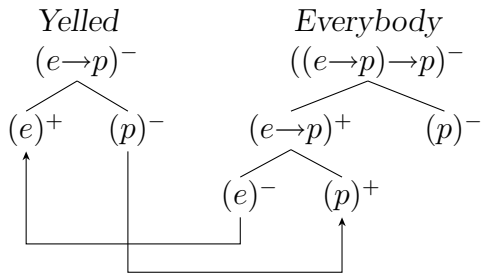
It should be evident that the arrows pointing up and to the left correspond to left-daughter links in the expression-trees, while the horizontal left-pointing arrows correspond to right-daughter links. The direction of axiom-links has also been reversed for consistent direction of flow. Although the horizontal left-to-right links might look like novel structure, note that this is not really the case because they are just the composite of the right-daughter-to-mother and mother-to-left-daughter links (which appear to have been the ones originally used in Lamarche's (1994) 'paths', the source for dynamic graphs). A somewhat more rigorous version of the proof-net-as-tree redisplay will be

[12] And they are also included if our frames include the standard obligatory positive.

presented in the appendix, but first it is necessary to extend the treatment to deal with more complex constructions, such as the positive implications that are used to analyse quantifiers.

First, a relatively simple example, *everybody yelled*, ignoring tense, correctly hooked up:

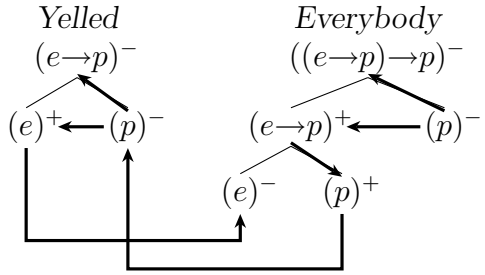
(13)



First, we extend the notion of expression-graph to structures like this, which contain positive implications as well as negative ones.

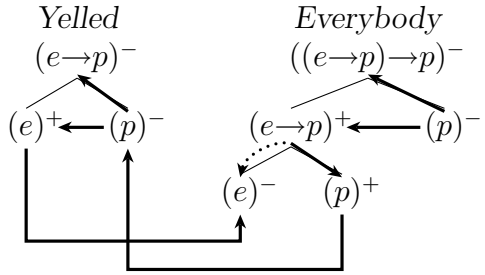
The dynamic graph rule for a positive implication is that it gets a link connecting it to its (positive) consequent, but *not* to its antecedent. So with the link-directionality of the expression-graph, we get this:

(14)



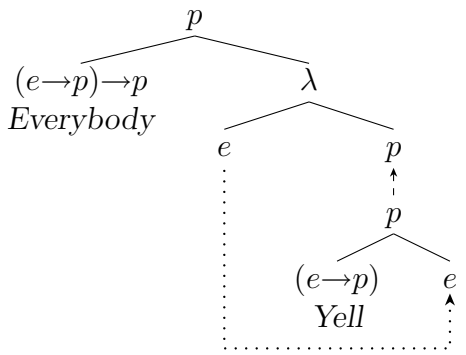
In terms of meaning, the positive implications correspond to lambda-expressions, but the dynamic graph and (so-far) expression-graph formats are insufficient for this, in that they don't do anything to explicitly represent binding. We can fix this by adding to the expression-graph an additional link, from a positive implication to its antecedent, for which we will use a curved dotted line:

(15)



With these conventions, (15) becomes (16) in the deplugged format of (11):

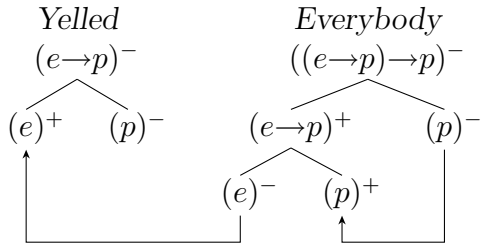
(16)



One difference between this and a conventional logical form is the presence of the dotted-line link rather than one variable appearing in two positions; we could clearly modify our procedure for rendering the proof-nets to yield this more conventional outcome instead. Another is that a lambda can bind only one position; we'll return to this issue later.

So far so good, but we can no longer put off dealing with the fact that not every way of connecting the literals in acordance with our rules so far leads to a sensible result. For example (17) below conforms to these rules, but clearly does not constitute a sensible reading:

(17)



If we consider what the dynamic graph, or expression-graph without the dotted link would look like, we note that it is (a) disconnected (b) fails to have a path connecting every full formula in the frame to the final output, while the sensible hookup of (13) has neither of these features.

There are in fact a tremendously large number of different ways to formulate the desired condition;[13] the following version is a straightforward consequence of de Groote (1999) (esp. Definition 4.1 and associated proofs):

(18) a. The dynamic graph provides a path from each member of the frame to the route (in the direction of flow).

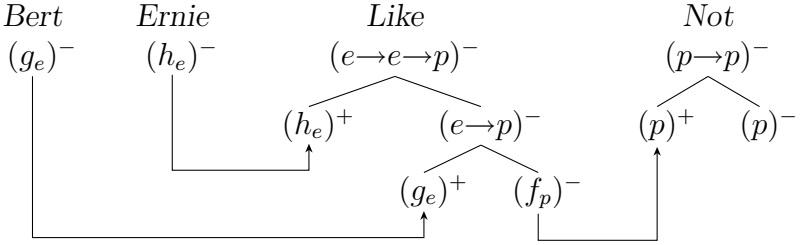
b. The dynamic path from the antecedent of a positive implication to the root must pass through the consequent of that implication.

This can also be formulated in terms of the expression-tree with dotted links removed.

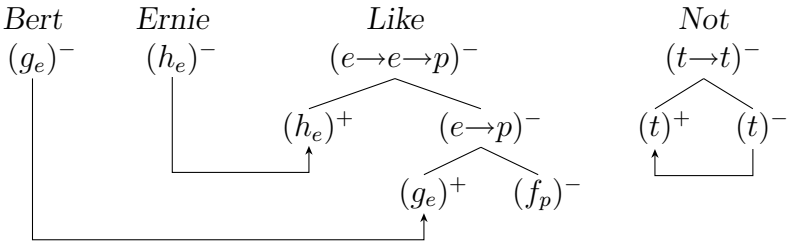
[13]See Crouch and van Genabith (2000) for a listing of some of them; Perrier (1999) and Moot (2002) ch 4 are also useful.

The criterion applies to rule out a variety of other pathological configurations, such as the (b) hookup of (19):

(19) a. Good:



b. Bad:



In general, (a) requires that the proof-net describe a sensible expression which includes all of the contributing meaning-components, while (b) requires that variables be properly bound. A final observation is that while a proof-net can't be recovered from its dynamic graph, since the information about variable-binding is discarded, it can be from its expression-graph with the dotted links included.

If the proof-nets/expression trees are to constitute a level of linguistic representation in LFG, this level needs a name, for which I'd suggest either 'semantic compositional structure', or 's-structure'. One name that would not be good is 'semantic structure', since the level doesn't have much to say about meaning as such, but simply prescribes how the meanings of parts of the sentence are to be assembled into meanings of the whole. The term 'semantic interpretive structure', or 'conceptual structure', might be used to a refer to a level where more information about the meaning itself was given.

2 Relating to the f-structure

Now that we've seen how to construe proof-nets as logical forms, the next issue to consider is their relationship to the rest of the linguistic structure. On the standard view, the glue derivation seems to float off to the side, bearing no clear relationship to anything else, although f-structure locations are mentioned on glue-sides. On the other hand, the notations do suggest some sort of linkage between the atomic formulas in the deduction, and substructures of the f-structure.

On the present view, this can be formulated quite nicely as a correspondence relation σ, which goes from the basically-typed nodes in proof-net to the substructures in the f-structure (essentially, in the opposite direction to ϕ). Hooking up frames into proof-structures then corresponds to the requirement that in order to be axiom-linked, atomic

formulas must have the same σ-correspondent.

It is straightforward to read out somewhat streamlined meaning-constructors in this way. For example (7, 8), repeated here for convenience, can be read as asserting that various items exist in the s-structure, as described on the glue-sides, with type-labelling and f-structure correspondents as indicated:

(7) $\begin{aligned} \textit{Like} &: (\uparrow \text{OBJ})_e \multimap (\uparrow \text{SUBJ})_e \multimap \uparrow_p \\ \textit{Bert} &: \uparrow_e \\ \textit{Ernie} &: \uparrow_e \end{aligned}$

(8) $\begin{aligned} \textit{Like} &: h_e \multimap g_e \multimap f_p \\ \textit{Bert} &: h_e \\ \textit{Ernie} &: g_e \end{aligned}$

That is, '$\uparrow_e$' can be read as saying: 'this s-structure node has type-label e, and its f-structure correspondent under σ is the same as the f-structure correspondent under ϕ' of the c-structure node I'm appearing under (in the annotated c-structure). Then, if the $\uparrow$ instantiates to, say g, the resulting literal 'g_e' says that the node's σ-correspondent is g (and type-label is e).

To illustrate the idea, if we had a meaning-constructor like this for *obviously*:[14]

(20) $\textit{Obvious} : (\text{ADJUNCTS} \in \uparrow)_p \multimap (\text{ADJUNCTS} \in \uparrow)_p$

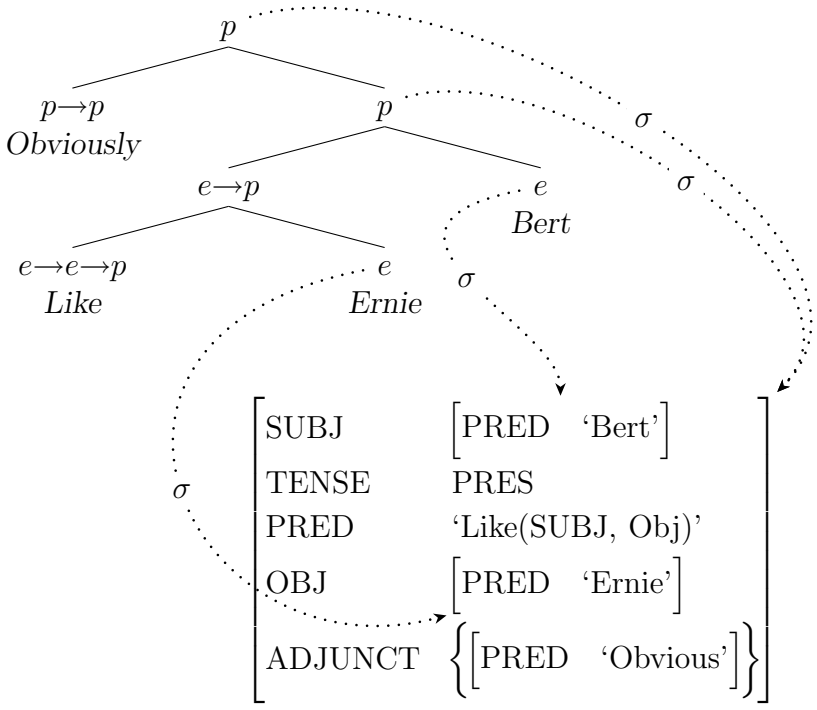
we'd get a semantic and f-structure pair like this for *Bert obviously likes Ernie*, with the σ-correspondence drawn in:

[14]This may be oversimplified (Dalrymple 2001:270-274), but note that adverbs don't seem to be subject to the kind of adverbial modification that makes the adjective constructors so complicated:

i) The obviously skillful lassoing of the calves impressed the spectators

ii) *The cowboy lassoed the calves obviously skillfully.

(21)



This illustrates the many-to-one character that σ, as used here, shares with ϕ.

We can extend this treatment to quantifiers, which we've used to motivate the correctness criterion, but so far not provided a full glue analysis of. The scope-ambiguity of quantifiers is standardly taken as motivation for some kind of extension of glue beyond propositional linear logic, to something involving some kind of quantification. In 'new glue', the standard proposal is a restricted version of Girard's System F, but Kokkonidis (2006) argues that we can get a simpler treatment by using first order glue quantification, treating the type information as predicates, the f-structure location as individuals. On this account, the instantiated constructors for *everybody seems to yell* might be:

(22) $\begin{array}{rl} \textit{Everybody} & : \forall H.((g_e \multimap H_p) \multimap H_p) \\ \textit{Sleep} & : g_e \multimap h_p \\ \textit{Seem} & : h_p \multimap f_p \end{array}$

If we linearly-universally (not LFG-ishly) instantiate H to h, we get the wide-scope reading, to f the narrow-scope one.[15] However this can be notationally streamlined by omitting the quantifier, and interpretating variables as implicitly universally quantified, so that (22) becomes:

(23) $\begin{array}{rl} \textit{Everybody} & : (g_e \multimap H_p) \multimap H_p \\ \textit{Sleep} & : g_e \multimap h_p \\ \textit{Seem} & : h_p \multimap f_p \end{array}$

[15]Note that the linear universal quantifier is properly glossed as *any*, not *all*, since it only gets instantiated once.

But with the idea of the σ-correspondence, we can go even further, and keep the linear logic propositional.

All we have to do is stipulate that the appearance of the one variable H in two positions means that these two positions correspond under σ to the same f-structure, without saying what it that f-structure is. The idea can be expressed more explicitly by reformulating the *everybody* constructor like this:

(24) $\mathit{everybody} \;:\; (X \multimap Y) \multimap Z$

$$\begin{array}{ll} \sigma(X) = g & \tau(X) = e \\ \sigma(Y) = \sigma(Z) & \tau(Y) = p \\ & \tau(Z) = p \end{array}$$

On this interpretation, H in (23) is being read in essence as an f-structural local variable.

So, if sharing of ADJUNCTS-values did turn out to cause a problem for (20), we could resolve it by reformulating it with a local variable as follows:

(25) $\mathit{Obvious} \;:\; H_p \multimap H_p, \uparrow \in (H\,\text{ADJUNCTS})$

Summarising the proposal, we reinstate a 'semantic projection' σ, but reverse its direction and change its functionality. σ will be a correspondence that relates the leaves/literals of frames to the substructures of the f-structures. Like ϕ, it will be a function, and also like ϕ, one that is neither one-to-one nor onto (the f-structure correspondents of expletives would be likely candidates for absence from the range of σ). In conjunction with the type-labelling function on the atomic types, σ constrains the combination of meaning-constructors on the basis that axiom-linked nodes must have the same type and σ-correspondent.

The glue quantifiers and variables can therefore be eliminated, their essential functionality being taken over by the correspondence relation σ. This amounts to treating them as meta-variables that can be LFG-instantiated by any f-structure. (However, in a computational implementation, it might be good to continue to treat them as linear quantification, which would function as an underspecified representation.[16])

The main motivation for setting up this formulation of glue is to make certain kinds of constraints easier to think about, but it is also worth saying a bit about its formal potential. This would appear to be that of Kokkonidis's (2006) 'first order glue', with only monadic type predicates, and no functional term-builders, such as Kokkonidis suggests for dealing with the 'dummy attributes' VAR and RESTR, which are found in standard glue analyses. Both restrictions follow from the fact that the only device that our present architecture ('correspondence-based glue', we might call it) has available to connect the semantic structures to the f-structure is a correspondence function from the leaves of the semantic type tree to the substructures of the f-structure, which is

[16](Moot 2002), ch. discusses proof-nets for universal quantification.

capable of representing the effects of monadic type predicates such as $e/1$ and $t/1$, but nothing more.

The unavailability of the dyadic predicates that Kokkonidis suggests isn't an empirical problem at this point, since no glue analysis using such things has yet been proposed, but the use of the dummy attributes in the analysis of quantification is well-established, as in constructors such as this for *every* and *dog* (using t instead of p in accordance with prevailing convention):

(26) $\begin{array}{rl} \textit{Every} & : ((_\sigma\uparrow \mathrm{VAR})_e \multimap (_\sigma\uparrow \mathrm{RESTR})_t) \multimap (_\sigma\uparrow_e \multimap H_t) \multimap H_t \\ \textit{Dog} & : (_\sigma\uparrow \mathrm{VAR})_e \multimap (_\sigma\uparrow \mathrm{RESTR})_t \end{array}$

The dummy attributes seem to be the main motivation for the 'semantic projection' of standard glue, which runs from f-structure to an additional feature-structural level where the dummy attributes are stashed; the motivation for putting them there is that they don't play any independent role in the f-structure.

Unfortunately, this projection doesn't seem to show any of the characteristics of a projection as discussed by Andrews and Manning (1999) (i.e. a distinctive pattern of attribute-sharing accross structural levels), so it is empirically inert. Kokkonidis suggests that the semantic projection could be dispensed with by treating the dummy attributes as functional term-builders in the glue component, so that alongside of for example $e(g)$, we could also have $e(\mathrm{var}(g))$. However, in spite of their popularity, the dummy attributes don't appear to actually do anything in the analyses, any more than the standard semantic projection itself does does. Indeed, everything works fine if the constructors are just:

(27) $\begin{array}{rl} \textit{Every} & : (\uparrow_e \multimap \uparrow_p) \multimap (\uparrow_e \multimap H_p) \multimap H_t \\ \textit{Dog} & : \uparrow_e \multimap \uparrow_p \end{array}$

One can certainly produce some strange grammars using this constructor and various other ingredients, but that is a feature of glue in general, since not that many constraints have been imposed on what can be done, and adding dummy attributes doesn't ameliorate this problem at all.

Correspondence-based glue is thus a simpler system with an architectural commitment to less powerful devices than alternate approaches to glue semantics for LFG, which gives it a reasonable claim to be preferred, until some phenomena requiring more powerful mechanisms appear. But the main motivation was not to be simpler than other approaches to glue, but to make it easier to explore structure-based constraints. But before looking at some candidates, we need to say a bit about the problems of bound anaphora.

3 Anaphora

Anaphora provides quite a range of problems for glue, and no consensus has yet emerged on how to deal with them. Two major issues for glue are:

(28) a. The involvement of cross-sentential relations in discourse anaphora

b. The general problem of 'resource deficit', in that anaphoric elements appear to 'borrow' resources from other locations in the grammatical structure.[17]

The current proposals for cross-sentential anaphora are Crouch and van Genabith (1999), Dalrymple (2001) and Kokkonidis (2005), the latter with a minor variant in Lev (2006a). They all involve either (a) an excursion from standard multiplicative linear logic on the glue-side (b) a departure from sentence-by-sentence logical forms on the meaning-side, or both at once. Since these features introduce substantial additional complexities for the use of proof-nets as logical forms, I won't consider them here, except for a very brief mention of DRT, but instead look at what appear to be the two standard proposals for bound anaphora, the first discussed by Lev (2006a), within the implicational fragment, the second by Asudeh (2004), requiring the addition of tensors. These two kinds of analysis use different techniques to address the problem (b) of resource deficit.

3.1 Lev 2006

This analysis lies within the implicational fragment of ILL, and uses a mechanism which is well supported for the treatment of reciprocals (Lev 2006b), although Lev himself ultimately prefers a variant of Kokkonidis's (2005) DRT-based treatment.

Like other glue analysis of anaphora, this analysis makes use of an ANTecedent attribute of the f-structure of the pronoun, whose value is the f-structure of the pronouns antecedent. The bound pronoun constructor is:

(29) $\lambda P.\lambda x.P(x)(x) \ : \ (\uparrow_e \multimap \ (\uparrow \mathrm{ANT})_e \multimap \ H_p) \multimap \ (\uparrow \mathrm{ANT})_e \multimap \ H_p$

Unlike our previous examples, we've specified some internal structure for the meaning-side.

Suppose that this is the constructor for, say *himself*, and that the sentence is *Ernie likes himself*, and that syntactic constraints on the ANT relation constraint it to be the same as the SUBJ-value of the clause. Then the instantiated constructors might be:

(30)
$$\begin{aligned} \textit{Ernie} \ &: \ g_e \\ \lambda P.\lambda x.P(x)(x) \ &: \ (h_e \multimap \ g_e \multimap \ f_p) \multimap \ g_e \multimap \ f_p \\ \textit{Like} \ &: \ h_e \multimap \ g_e \multimap \ f_p \end{aligned}$$

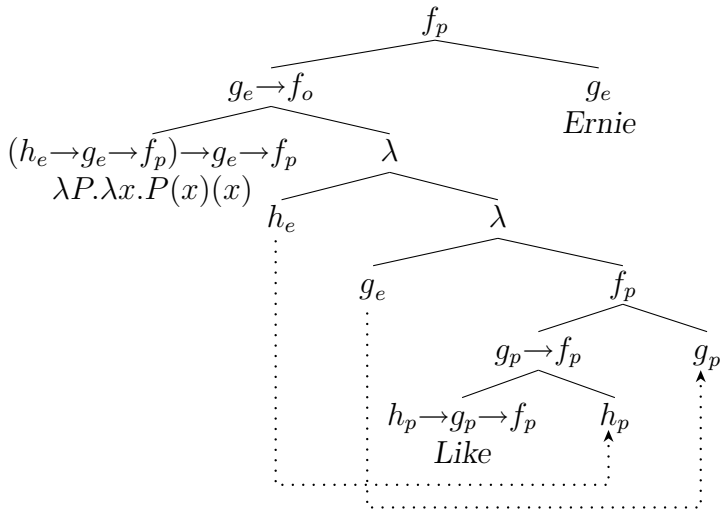
We can represent how these are supposed to be assembled rather compactly by drawing the links directly between the literals:

[17] See Asudeh (2004) for an extensive discussion of issues of resource surplus and deficit.

(31)
$$\begin{array}{rl} \textit{Ernie} & : \; g_e \\ \lambda P.\lambda x.P(x)(x) & : \; (h_e \multimap g_e \multimap f_p) \multimap g_e \multimap f_p \\ \textit{Like} & : \; h_e \multimap g_e \multimap f_p \end{array}$$

The (plugged) expression-tree format version of this will be:

(32)



This expresses the correct meaning $\textit{Like}(\textit{Ernie}, \textit{Ernie})$, once the β-reductions involving the reflexive meaning are performed (and might be helpful in producing an analysis of the 'sloppy identity' phenomenon in the interpretation of *John washed himself and so did Bill*), but note that from the point of view of glue alone, the logical form is just (assuming Montague's representational convention of of writing $P(x, y)$ for $P(y)(x)$, etc.):

(33) $\textit{Self}(\textit{Ernie}, \lambda y.\lambda x.\textit{Like}(x, y))$

This is because linear logic on its own can't just double a resource, since the rule of Contraction isn't freely available.

So to get a logical form which fully expresses the coreference between subject an object positions, we must in effect do β-reductions in a more liberal logic where contraction is available. See de Groote and Retoré (1996) and Morrill (2005) for a way of doing this while continuing to use proof-nets, although some work would need to be done to adapt these ideas from the context of Categorial/Type-Logical Grammar to LFG+glue.[18]

Rather we'll consider another problem, which is that the *Self* operator leads to a spurious structural ambiguity at the glue-level in the presence of other operator such as modals, negation, etc. For example *Ernie probably likes himself* would be either:

[18]The main issue being type-mismatch: de Groote and Retore convert the typying of the syntactic nets to that of semantic ones, by replacing certain basic types to ones involving the (linear logic) exponential; Morrill appears to omit explicit discussion of this step.

(34) a. $\mathit{Probable}(\mathit{Self}(\mathit{Ernie}, \lambda yx.\mathit{Like}(x, y)))$

b. $\mathit{Self}(\mathit{Ernie}, \lambda yx.\mathit{Probable}(\mathit{Like}(x, y)))$

For reciprocals, similar ambiguities are sometimes genuine (Lev 2006b), but in the case of reflexives, there seems to be no hint of either an intuitive structural or interpretational ambiguity. This might simply be because the two analysis β-reduce to the same thing, but it's still a point to worry about.[19]

The good features of this analysis are:

(35) a. it fits into the purely implicational fragment of Intuitionistic Linear Logic.

b. it uses a form of constructor that is independently motivated for reciprocals (Lev 2006b).

The bad are:

(36) a. it introduces a structural ambiguity that corresponds to no perceived ambiguity.

b. it seems to have rather dim prospects for being applicable to cross-sentential anaphora.

c. as we'll discuss in the next section, it's hard to imagine how it could deal with split antecedents for bound anaphora.

3.2 Asudeh (2004)

The next analysis we'll consider avoids the spurious ambiguities, but requires that we depart from the implicational fragment by introducing tensors, which also makes it harder, although not impossible, to read proof-nets as being somewhat conventional logical forms. The tensor, written '$\otimes$', may be regarded as 'splitting' a resource into left and right streams, which can flow independently for a while, but must eventually both wind up as contributors to a single result (so that neither part can just be thrown away). For this to lead to a sensible analysis of anaphora, this resource needs to be on the meaning-side of a pair formed by doubling an input nominal meaning, so that the meaning-constructor for a pronoun looks like this:

(37) $\lambda x.[x, x] \ : \ (\uparrow \mathrm{ANT})_e \multimap \ ((\uparrow \mathrm{ANT})_e \otimes \uparrow_e)$

What this constructor does is:

(38) a. grab the meaning of the pronoun's antecedent (presumed accessible by an ANTecedent function).

[19] And note that, unlike some other spurious ambiguities noticed by Lev, this one can't be suppressed by requiring the reflexive to take its scope at the lowest possible c-structure node.

b. make two copies.

c. put one back at the original location of the antecedent.

d. deposit the other at the location of the pronoun.

Asudeh's presentation is in terms of the Natural Deduction formulation of glue, in which meaning-terms are built simultaneously with the construction of the deduction, and tensors are managed by a rather complex $\otimes$-elimination rule that uses the somewhat intimidating '**let**' meaning-term construction.

No purpose would be served by recapitulating Asudeh's lucid presentation of the analysis (see also Lev's (2006a) discussion), so I will merely observe that for a sentence such as *Bo probably fooled himself* (c.f. Asudeh 2004:139), we get the following two glue meaning-terms:

(39) a. $\mathit{Probable}(\textbf{let}\ \mathit{Self}(\mathit{Bo})\ \textbf{be}\ x \times y\ \textbf{in}\ \mathit{Fooled}(y, x))$

b. $\textbf{let}\ \mathit{Self}(\mathit{Bo})\ \textbf{be}\ x \times y\ \textbf{in}\ \mathit{Probable}(\mathit{Fooled}(y, x))$

As with the earlier analysis by Lev, both of these will β-reduce to the same thing, provided we know that $\mathit{Self} = \lambda x.[x, x]$, by these stages for (b), and comparably for (a):

(40) a. $\textbf{let}\ \lambda x.[x, x](\mathit{Bo})\ \textbf{be}\ x \times y\ \textbf{in}\ \mathit{Probable}(\mathit{Fooled}(y, x))$

b. $\textbf{let}\ [\mathit{Bo}, \mathit{Bo}]\ \textbf{be}\ x \times y\ \textbf{in}\ \mathit{Probable}(\mathit{Fooled}(y, x))$

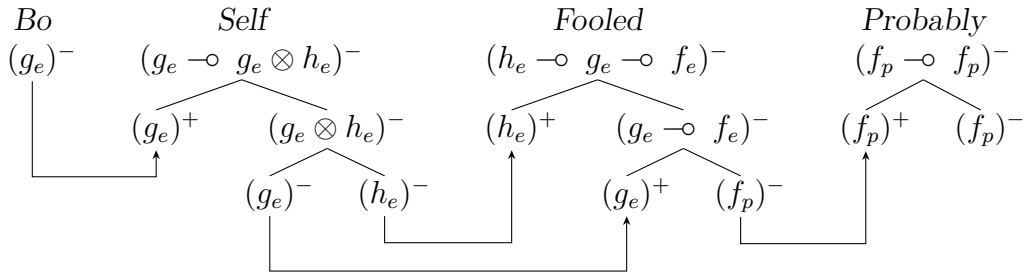
c. $\mathit{Probable}(\mathit{Fooled}(\mathit{Bo}, \mathit{Bo}))$

Using proof-nets, things look a bit different, and the structural ambiguity disappears. Tensors are introduced into proof-nets by adding positive and negative 'homogeneous tensor links' (terminology from de Groote 1999). Positive tensors are trivially eliminable, while negative ones can be smoothly introduced into proofnets by adding this polarity rule:

(41) The right and left components of a negative tensor are both negative (ditto for positives, replacing 'negative' with 'positive' throughout).

We can now represent the proof-net for *Bo probably fooled himself* as follows (using the usual linear logic implication symbol '$\multimap$' rather than '$\rightarrow$' for the implications, for typographical congruity with the tensor):

(42)



If we use the **let** construction in our logical forms, we encounter a difficulty here, in that the existence of the two structures in (39) doesn't fit well with the single net of (42).

This problem can be addressed by taking a different approach to the proof-terms, following Moot (2002:19) in using the 'projections' π^1 and π^2 to acess the first and second members. On this account, the proof-term corresponding to the network (42) would be:

(43) $\mathit{Probable}(\mathit{Fool}(\pi^1(\mathit{Self}(\mathit{Bo})), \pi^2(\mathit{Self}(\mathit{Bo}))))$

This also β-reduces to (40c), but from only one source rather than two.

In spite of this improvement, the analysis shares an essential feature with the treatment with **let** (as well as Lev's analysis from the previous subsection), that we have in the glue-derivation some structures which can't be simplified within glue itself, in this version, the two projections, which have the additional annoying feature of both applying to the same complex substructure, *Self*(*Bo*).

The irreducibility is unavoidable, since glue lambda's are linear, so we need to work in a more liberal system in order to get the desired final result (40c), but the duplication is just an artifact of trying to use a strictly linear representation. If we construct an expression graph in the obvious way, running links from the two components of the negative tensor back to the whole, we can see that π^1 represents what we see by entering the *Self*(*Bo*) substructure through the left link, π^2 from the right. In a linear representation, one structure can't sit in two positions, so we have to make a copy, while in a graphical representation such as a proof-net we don't have this problem. So at this point the paralellism between the proof-nets and conventional logical forms breaks down, but I think it's clear that that is more a problem with the latter than the former: if we're interested in modelling some kind of cognitive reality, which is what linguistics is about, the use of linear formats may be practically convenient, but is certainly not fit to be a matter of principle.

And, if we really need a linear representation for some reason, the correctness criterion guarantees that the projection technique will work, as discussed in the proofnet literature.

One advantage of this analysis over the previous one is that if we use proof-nets, the spurious ambiguity with *probably* disappears.[20] Another is that it articulates with a worked-out treatment of resumptive pronouns and certain other cases of 'resource surplus', as discussed by Asudeh, which is not the case for the previous analysis, so far at least.

A more striking empirical treatment is that it can provide a reasonable treatment of cases of split antecedents with bound anaphora, such as:

(44) a. Every professor$_i$ told a student$_j$ that they$_{\{i,j\}}$ would co-author a paper

[20]It could be made to disappear with natural deduction as well, but the norm is for distinct natural deductions to have different proof-terms, so the **let** formulation seems better for these systems.

b. Every professor$_i$ told a student$_j$ that the head of department$_k$ expected that they$_{\{i,j,k\}}$ would co-author a paper

c. John$_i$ gave Mary$_j$ a picture of themselves$_{\{i,j\}}$ standing on top of the mountain

These can be accomodated with a meaning-constructor for plural pronouns that takes the form of a schema, similar to what was proposed by Beryozkin and Francez (2004) for what might be reasonably described as 'split functional control'.

For this, we will need to allow plural pronouns to have a set-valued antecedent relation that we'll call ANTECEDENTS attribute, each member of which is the f-structure of a distinct antecedent of the pronoun. It will also be useful to have a notation to refer to an n-ary tensor whose σ-correspondents are the members of the value of this function, for which $\otimes(\uparrow \text{ANTECEDENTS})_e s$ would seem a reasonable choice.

We can now write the following glue-side:

(45) $\otimes(\uparrow \text{ANTECEDENTS})_e \multimap \uparrow_e \otimes (\otimes(\uparrow \text{ANTECEDENTS})_e)$

For the meaning-side, we'll use the notation $\oplus C$ to mean the i-sum (in the sense of Link 1998) of the components of the n-ary tensor C. The entire constructor can then be written as:

(46) $\lambda C.[\oplus C, C] \ : \ \otimes(\uparrow \text{ANTECEDENTS})_e \multimap \uparrow \otimes (\otimes(\uparrow \text{ANTECEDENTS})_e)$

At this point, I can't see any way of adapting the analysis of the previous section for multiple antecedents, which is interesting in light of the fact that these are completely unavailable for reciprocals, for which this analysis seems quite appropriate:

(47) a. John and Mary gave Bill pictures of each other

b. Bill gave John and Mary pictures of each other

c. *John gave Mary pictures of each other

That's a good point; some negative ones are:

(48) a. the spurious ambiguity is still there with Natural Deduction (since proof-terms are supposed to be in 1-1 correspondence with natural deductions, which demand a genuine tree structure, so that **let** rather than the projections are the appropriate proof-term builders).

b. no hint at all for dealing with sloppy identity (not a really serious problem until there is a glue analysis of identity-of-sense anaphora).

c. It is unclear how to extend the analysis to deal with cross-sentential anaphora.

3.3 Kokkonidis 2006

A possible resolution to these problems is provided by the DRT-based analysis of Kokkonidis (2005), and a minor variant suggested by Lev (2006a). These analysis hand over the responsibility for dealing with referential resources to a DRT component that interfaces with the meaning-side, so that as far as glue is concerned, all NPs look either like quantifiers (of type $(e{\rightarrow}p){\rightarrow}p$) or type e constants. The s-structure for *a boy washed himself* would then be essentially:

(49) $\mathit{Some}(\mathit{Boy}, \lambda x.\mathit{Self}(\lambda y.\mathit{Wash}(x, y)))$

with the DRT mechanisms responsible for this coming out essentially equivalent to:

(50) $\mathit{Some}(X, \mathit{Boy}(X), \mathit{Wash}(X, X))$

A thorough treatment of this would go rather beyond our aims here. Something to consider is Jäger's (2005) proposal (in the TLG framework) to enrich the composition logic with a limited form of Contraction. Perhaps stuff could be copied via antecedent relations but not otherwise? This seems to me to be sort of like what DRT does.

4 Constraints

Something about node ordering and 'Outscopes'. Or maybe not, 20pp is a reasonable length, perhaps.

A From Proof-net to Expression-Tree

We start with an implicative proof-net, and begin at the sole unlinked negative terminal, and move through the tree adding links labelled s, l, r (sole, left, right). The procedure that does the work will be called 'Relink'. The method is described as follows:

(51) At the sole unlinked negative leaf n, run Relink(n), defined as follows:

Relink(n):

a. If n is the root of a type-tree (i.e. a node with a semantic label), or the antecedent of a positive implication, stop.

b. If n is axiom-linked to the node m:

i) Add an s-link from c to m

ii) Relink(m)

(This step gives us one of the dotted lines from a mother to a daughter-position).

c. If n is the consequent of a negative implication b with antecedent a:

i) Add an r-link from n to a
ii) Add an l-link from n to b
iii) Relink(a)
iv) Relink(b)

d. If n is a positive implication with antecedent a and consequent b:
i) Add an l-link from n to a
ii) add an r-link from n to b
iii) Relink(b)

(These are the nodes labelled with λ in the expression-tree format).

Since the procedure does nothing but add links on the basis of local conditions, in such a way that the original proof-net can be recovered from the new one on a similar basis, it can be seen as merely looking at the proof-net from a different angle.

The extension to positive and negative tensors is trivial.

Bibliography

Andrews, A. D. 2006. Semantic composition for NSM, using LFG + Glue. Proceedings of ALS2005, URL: `http://www.als.asn.au/`.

Andrews, A. D. to appear. Generating the input in OT-LFG. In Grimshaw, Maling, Manning, and Zaenen (Eds.), *Architectures, Rules, and Preferences: A Festschrift for Joan Bresnan*. Stanford CA: CSLI Publications. URL: `http://arts.anu.edu.au/linguistics/People/AveryAndrews/Papers`.

Andrews, A. D., and C. D. Manning. 1999. *Complex Predicates and Information Spreading in LFG*. Stanford, CA: CSLI Publications.

Asudeh, A. 2004. *Resumption as Resource Management*. PhD thesis, Stanford University, Stanford CA. `http://http-server.carleton.ca/~asudeh/` (viewed Oct 2006).

Asudeh, A., and R. Crouch. 2002. Coordination and parallelism in glue semantics: Integrating discourse cohesion and the element constraint. In *Proceedings of the LFG02 Conference*, 19–39. CSLI Publications. URL: `http://csli-publications.stanford.edu`.

Beryozkin, G., and N. Francez. 2004. The "lost" reading of control sentences and plural semantics in glue. In *Proceedings of the LFG02 Conference*, 80–100. CSLI Publications. URL =`http://csli-publications.stanford.edu`.

Crouch, R., and J. van Genabith. 1999. Context change, underspecification, and the structure of glue language derivations. In Mary Dalrymple (Ed.), 117–189.

Crouch, R., and J. van Genabith. 2000. Linear logic for linguists.
URL: `http://www2.parc.com/istl/members/crouch/`.

Dalrymple, M. (Ed.). 1999. *Syntax and Semantics in Lexical Functional Grammar: The Resource-Logic Approach*. MIT Press.

Dalrymple, M. 2001. *Lexical Functional Grammar*. Academic Press.

de Groote, P. 1999. An algebraic correctness criterion for intuitionistic multiplicative proof-nets. *TCS* 115–134. URL: `http://www.loria.fr/~degroote/bibliography.html`.

de Groote, P., and C. Retoré. 1996. On the semantic reading of proof-nets. In G. G.-J. Kruijff and D. Oehrle (Eds.), *Formal Grammar*, 57–70, FOLLI Prague, August. URL: `citeseer.ist.psu.edu/degroote96semantic.html`.

Jäger, G. 2005. *Anaphora and Type Logical Grammar*. Springer.

Kokkonidis, M. 2005. Why glue a donkey to an f-structure when you can constrain and bind it instead. In M. Butt and T. King (Eds.), *Proceedings of LFG 2005*. URL: `http://csli-publications.stanford.edu`.

Kokkonidis, M. 2006. First order glue. *Journal of Logic, Language and Information*. to appear; available on request according to `http://users.ox.ac.uk/~lina1301/`.

Lamarche, F. 1994. Proof nets for intuitionistic linear logic 1: Essential nets. Technical Report, Imperial College.

Lev, I. 2006a. Anaphora in glue semantics. URL: `http://www.stanford.edu/~iddolev/papers/gmemo_anaphora.pdf`.

Lev, I. 2006b. On the syntax-semantics interface of overt and covert reciprocals. URL: `http://www.stanford.edu/~iddolev/papers/ling233b_quants.pdf`.

Link, G. 1998. *Algebraic Semantics in Language and Philosophy*. Stanford CA: CSLI Publications.

Moot, R. 2002. *Proof-Nets for Linguistic Analysis*. PhD thesis, University of Utecht. URL: `http://www.labri.fr/perso/moot/`.

Morrill, G. 2005. Geometry of language and linguistic circuitry. In C. Casadio, P. J. Scott, and R. Seely (Eds.), *Language and Grammar*, 237–264. CSLI Publications.

Perrier, G. 1999. Labelled proof-nets for the syntax and semantics of natural languages. *L.G. of the IGPL* 7:629–655. URL: `http://www.loria.fr/~perrier/papers.html`.