Semantic Interpretation by CC Functors for LFG Glue Semantics

version 2

Avery D Andrews

ANU, July 2010

The 'glue semantics' that has been developed for Lexical-Functional Grammar (LFG)[1] is based on the idea of using linear logic proofs to effect the semantic assembly of the meaning of a sentence from the meanings of its parts. The production of the meanings or model-theoretic interpretations themselves is standardly achieved by labeling the nodes of the deduction with terms in a combination of linear and non-linear lambda-terms, raising questions of exactly what is the relationship between these two systems, and suggesting that there might be a somewhat arbitrary symbolic intermediary sitting between the glue proofs and the actual meanings.

Here I will show how, if the meanings themselves are inhabitants of a closed cartesian category (CCC), as is often (but not always) the case, we can make the relationship between the glue proof and the assembly of the actual meanings more principled, by implementing it with a closed cartesian (CC) functor from a CCC based on the glue proof to the CCC that the model-theoretic interpretations reside in, which, furthermore, can go via a functor generated by the semantic types and the lexicon. Furthermore, if desired, this functor can be factored through some intermediate levels which, from a mathematical point of view, exist independently, and possibly have empirically relevant properties.

I will first give some background for LFG, to provide an account of the context in which the ideas are meant to function. A real explanation of the category theory on the other hand, developed in Lambek and Scott (1986:41-60),[2] henceforth LS, is probably not a realistic prospect for inclusion in a

[1] See Asudeh and Toivonen (2009b) for a recent overview, including guide to further literature, and Asudeh (2004), Kokkonidis (2008), Lev (2007) and Andrews (2008) for discussion of how glue-semantics fits into LFG.

[2] For which Barr and Wells (1999b), Barr and Wells (1999a) provide the necessary background (the introductory sections of LS itself are rather terse, and furthermore cover a lot of content that isn't required for what we're doing here). Barr and Wells' online notes seem to be particularly well-focused on things that are likely to be useful for linguistics. Pierce (1991) is also a good short introduction, which might be almost enough in combination

reasonably sized paper. But appendix A spells out the basics, although doesn't rehearse all of the needed proofs, which can be found in LS.

But a general remark is perhaps worth making: the theory of CCCs can be seen as a sort of 'algebraicization' of certain aspects of set theory that are especially relevant for describing semantic composition, in the same general way as secondary school algebra is an algebraicization of certain aspects of arithmetic that are useful for figuring out which if any numbers satisfy various kinds of constraints. So the substance of the CCC ideas we will be using is such an algebraicization of things that one would learn about in introductory courses on Math for Linguists and Formal Semantics. Some concepts from proof-theory as such are also used, which are usefully surveyed in Crouch and van Genabith (2000). Restall (2000) is another useful source.

I conclude the introduction with a brief remark on motivation. LFG+glue can be regarded as a combination of a 'linguists' syntax' (certainly not the only one) and a 'logician's semantics', essentially the same as employed by Type-Logical Grammar and related theories. So why not 'go logical' all the way? The answer is that, as a 'linguists' syntax', LFG has a substantial track record in dealing with a typologically diverse range of descriptive problems, such as syntactic ergativity (Manning 1996), case-marking in Australian Languages (Simpson 1991, Nordlinger 1998), case-marking and agreement in Icelandic (Andrews 1982, 1990, Zaenen et. al. 1985), and may others, which even very recent logically based theories, such as for example Pollards' 'Convergent Grammar'[3] do not seem to address.[4] This is not supposed to imply anything about the ultimate merits of the different approaches, but simply to argue that LFG+glue might be a useful avenue to explore.

1. LFG+Glue

LFG is a grammatical theory that is largely based on the idea of using annotations on phrase-structure rules to connect phrase-structures ('c-

with appendix A.

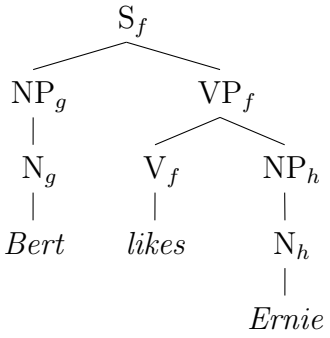
[3] `http://www.ling.ohio-state.edu/~scott/cvg/`

[4] But there have been some interesting attempts to put LFG itself on a more type-logical footing (Johnson 1999, Oehrle 1999, Kokkonidis 2007d). Kokkonidis' 'TL-LFG' project seems to be developing more persistently than the others, but there is still a great deal of descriptive material for it to work through in order to become a replacement for LFG+Glue.

structures') to more abstract 'functional structures' ('f-structures') which support straightforward accounts of many phenomena of grammar and semantic interpretation. It is a considerable surprise that such a simple (and computationally implementable) mechanism can handle so many of the kinds of syntactic phenomena that used to be seen as providing strong evidence for Transformational Grammar, which turned out to be computationally hopeless in its original form.

Here is a typical c-structure f-structure pair:

(1) a.



b.

$$f:\begin{bmatrix} \text{SUBJ} & g:\begin{bmatrix}\text{PRED} & \text{'Bert'}\end{bmatrix} \\ \text{TENSE} & \text{PRES} \\ \text{PRED} & \text{'Like(SUBJ, OBJ)'} \\ \text{OBJ} & h:\begin{bmatrix}\text{PRED} & \text{'Ernie'}\end{bmatrix} \end{bmatrix}$$

The c-structure (a) represents the linear order of words, and also their grouping into phrases, as determined by phrase-structure rules and schemas, while the subscripts represent a mapping (one-to-many, not in general onto) of c-structure nodes into parts of the f-structure (b), where various kinds of grammatical information are represented in a manner that is conceptually quite similar to traditional grammar, and producible by a relatively modest form of augmentations to the phrase-structure rules. For example the first NP is characterized as the 'subject', the second as the 'object', and information about grammatical properties such as tense, gender, number, case, etc. can be provided in the same format. Both the c-structure and the mapping are described by annotated phrase-structure rules and lexical entries, which for example say such things as 'the first NP under the S has as its f-structure

correspondent the f-sub-structure that is the SUBJ-value of the f-structure correspondent of the S', or, 'the f-structure correspondent of the VP under the S is the same as the f-structure correspondent of the S'.

The f-structure level and the nature of its relationship to c-structure enables a very diverse range of language structures to be described, for example ones such as Latin or Russian with highly variable word-order and rich case-marking, as well as ones like English, with relatively fixed word order and very little (or no) case-marking. The notion of f-structure in particular allows concepts such as 'subject' and 'object', originally worked out for Latin and Greek, to be usefully applied to English, while more typologically diverse languages, where the existence of subjects and objects is questionable, can treated by modifying assumptions about what the grammatical relations are and how they behave (Manning 1996). For more on the architecture, see for example Kaplan (1995) and Asudeh (2006).

Although f-structures handle certain aspects of semantic interpretation quite well, such as expressing the 'semantic role' structure of 'who is doing what to who' (except when 'serial verbs' are involved, where things get problematic), there are other important areas where they have major problems. The reason is that f-structures tend to 'flatten out' grammatical structure in ways that are very convenient for some purposes, such as marking tense on a constituent such as a verb, buried inside a VP, or case on a noun, buried inside an NP, but creates difficulties for many classic problems of semantic interpretation, such as differentiating the two readings of sentences such as *everybody loves somebody*, or *everybody didn't leave*, which would have a single f-structure of the same general form as (1b).

After the various attempts during the 1980s to address the problems of semantic interpretation for LFG, glue semantics was developed in the 1990s, as an application of (a quite rudimentary fragment of) Girard's linear logic.[5] A main virtue of linear logic is that it enforces the 'use everything, but only once' principle that seems to be the default for semantic interpretation of natural language (you can't construe a denial as a confession by ignoring the negative or interpreting it twice), without relying on the grammatical structure being a tree, as most semantic interpretation schemes used in linguistics

[5]Most of the significant work from this period appears in Dalrymple (1999).

do. This is essential because f-structures in general are not trees, definitely including re-entrancies, and possibly even cycles.

One basic idea of glue is that the words and possibly other meaningful elements (formatives within words, and perhaps certain phrase-structure configurations) of the sentence provide a collection of 'meaning-constructors', which are the semantic contributions from which the meaning of the whole utterance is to be assembled, together with a specification of how the assembly is to be constrained by the syntax. For a first approximation, the meaning-constructors can be thought of as pairings of an actual meaning with a type, which specifies both the semantic type of the meaning, and information about how it is connected to the syntax. The types are constructed from some basic types and at least a type-constructor for function application, and probably another for pair-formation (in principle, there might be more, but that's all that seems to be currently encountered in practice). A source of notational confusion is that because the logic of assembly is linear, the product will be a tensor as far as glue itself is concerned, but will behave like a cartesian product in the semantics. I'll try to keep the notation appropriate to the immediate circumstances.

Therefore, if we have semantic types E for 'entity' and P for 'proposition',[6] and temporarily ignore syntax, the meaning-constructors for *Bert likes Ernie* might be (assuming the usual convention of omitting rightmost parentheses with implications):

(2)

E	E	$E \multimap E \multimap P$
Bert	*Ernie*	*Like*

Note that the verb is treated as 'curried'; linguists find various motivations[7] for supposing that the arguments of verbs are organized in a hierarchy with 'least active' corresponding to 'applied first' or 'bonded tighter' in a curried scheme. So the first E of *Like* represents the 'Likee', the second, the 'Liker'.

[6] It is standard to assume at least these two basic types, although e and t are the commonest choice for the symbols. Note however that more basic types are often envisioned (e.g. Jackendoff (1990), Lev (2007)), and there might be only one (Partee 2006). We use 'proposition' in order to foreshadow the possibility of using more than just extensional model theories.

[7] Such as the 'Thematic Role Hierarchy' of Jackendoff (1972), and the idiom-structure argument of Marantz (1984).

The motivations for currying are widely but not universally accepted; dissidents can and sometimes do use products/tensors rather than curries (assuming the usual implication that product-like operators bind more tightly than implications):

(3)
E	E	$E \otimes E \multimap P$
Bert	*Ernie*	*Like*

With tensoring of the arguments, the more active one would conventionally be listed first.[8] Since the logic of combination is (commutative) linear, the constructors form a multiset, so that relative order is not significant, but multiplicity of occurrence is.

There are two possible ways of combining the constructors of (3), which correspond to different linear logic deductions of a type P conclusion from them. These can be conveniently represented by proof-nets,[9] where the type P final conclusion is separated by a turnstyle from the assumptions, and the curved lines are 'axiom links', which represent applications of the identity axiom in the Gentzen sequent calculus:[10]

(4) a.
E	E	$E \multimap E \multimap P$	$\vdash$	P
Bert	*Ernie*	*Like*		

b.
E	E	$E \multimap E \multimap P$	$\vdash$	P
Bert	*Ernie*	*Like*		

Here we have supplemented the constructors (premises) with an additional formula of propositional type (the conclusion) so as to form a sequent, and paired literals of matching types to represent a (linear) proof of the conclusion from the premises.

Unfortunately, not any old type-matched pairing of literals represent a valid proof; for this to be the case, the 'Correctness Criterion' for proof-nets must be satisfied, which has many different formulations. The algebraic formulation of de Groote (1999) is the one with the closest relationship to what is being done here.

[8] For roughly the same kinds of reasons that motivate the layering of arguments in the curried scheme, following a correspondence of linear 'earlier' with layered 'more outer'.

[9] To which Moot (2002) provides an excellent general introduction oriented towards the needs of linguistics.

[10] Discussed in Crouch and van Genabith (2000).

If the types specify only semantic information, the proof-net rules will tell us what coherent meanings we can build from a given pile of meaning-pieces, a notion that has been of explicit interest in linguistics at least since Klein and Sag (1985). But for an adequate account of NL meaning-assembly, we need also to account for how grammatical structure limits the assembly possibilities. In LFG, the way in which the constructors are introduced provides them with links to the f-structure, which can be regarded as a second set of types, parallel to the semantic ones (notated with f-structural location subscripts on the semantic types, or vice versa). For example, the lexical (uninstantiated)[11] constructor associated with the verb *Like(s)* would be along the lines of:

(5) $\mathit{Like} : (\uparrow \mathrm{OBJ})_E \multimap (\uparrow \mathrm{SUBJ})_E \multimap \uparrow_P$

Here the '↑' means 'the f-structure correspondent of the c-structure node I (an occurrence of a form of the verb *like*)am being introduced under', '(↑OBJ)' means 'the OBJ-value of the f-structure correspondent of the node I am being introduced under', etc. On the basis of this notation, the material to the left of the colon is called the 'meaning side', since it's supposed to indicate what the meaning is, while that on the right is called the 'glue side', since it contains the assembly instructions.

Then, if the sentence is *Bert likes Ernie*, the constructors of (2) become 'instantiated', perhaps as (6) below, given the f-structure of (1) above:

(6) $\underset{\mathit{Bert}}{g_E} \quad \underset{\mathit{Ernie}}{h_E} \quad \underset{\mathit{Like}}{h_E \multimap g_E \multimap f_P}$

There is furthermore a requirement that in order to get a declarative sentence meaning, these premises must deliver a conclusion of type f_P, where f is the label of the entire f-structure. This becomes the consequent of the sequent, so that (6) becomes:

(7) $\underset{\mathit{Bert}}{g_E} \quad \underset{\mathit{Ernie}}{h_E} \quad \underset{\mathit{Like}}{h_E \multimap g_E \multimap f_P} \vdash f_P$

Given this, the (b) linking of (4) becomes the only possibility. In this particular example, the f-structure information alone is sufficient to determine a

[11] That is, not associated with any particular occurrence in a grammatical structure.

unique linking, but there are also cases where the semantic types are needed to correctly constrain linking,[12] and where multiple assemblies are possible for a single set of constructors, as discussed in the literature.

2. Languages, Interpretations, and Functors

Our first observation is that a lexicon of typed meanings like those discussed above gives rise to, and in effect just is, a particular kind of free CCC as presented by LS and in appendix A. First, the system of semantic types is a collection of CCC objects, to which we can add a terminal object $\mathbf{1}$ plus the standard CCC arrows and operations on arrows to get the free CCC we'll call $\mathscr{T}$ generated by the basic types. Then, for each (uninstantiated) meaning-constructor in the lexicon, we adjoin to $\mathscr{T}$ an indeterminate $a{:}\mathbf{1} \to A$, where A is the semantic type of the constructor. The indeterminates sourced in this manner will be called 'lexical indeterminates' (it doesn't matter what order they are adjoined in), and the result of adjoining them the category $\mathscr{L}$.

The category $\mathscr{L}$ provides us with an infinite supply of what might be called 'formal propositions', arrows from $\mathbf{1}$ to P, that are essentially finitistic objects. For example if we have lexical indeterminates $\mathit{Uther}{:}\mathbf{1} \to E$ and $\mathit{Sleep}{:}\mathbf{1} \to [E \Rightarrow P]$, we then have this formal proposition (see appendix A for the notation):

(8) $\mathbf{ev}_E^{[E \Rightarrow P]} \langle \mathit{Sleep}, \mathit{Uther} \rangle$

The next observation is that the standard method of setting up a model-theoretic interpretation for a language in set-theory is equivalent to producing a CC functor[13] $I_{\mathscr{L}}$ from $\mathscr{L}$ into (a small subcategory of) the CCC **Sets**. More generally, an interpretation for a lexicon will be a functor from $\mathscr{L}$ into some CCC $\mathscr{S}$, which we will call the 'semantics category' (so, in principle, it doesn't have to be (a subcategory of) **Sets**).

In detail, a conventional specification of an interpretation would:

(9) a. Specify operations on sets as images of the product and exponential operations on types/formula-building operations (standard cartesian product and function space being the usual choices).

[12] Dalrymple p.c., discussed in Kokkonidis (2008).

[13] In linguistics, it's probably OK to restrict attention to strict ones.

b. with the obvious implicit choices for the projection and evaluation arrows, and pairing and currying operations on arrows.

c. Specify specific sets to serve as the images of the basic types, standardly, a 'universe of discourse' (often notated A) as the image of E, and a set of 'truth-values' (or perhaps 'possible worlds') as the image of P.

d. Specify specific members of sets of the appropriate types as interpretations of the meaningful items.

However, if we construe **Sets** as a CCC, by making specific choices for the CCC apparatus (including a terminal object and arrows, not mentioned in (9)), then the work of (a-b) is done by imposing the requirement that an interpretation be a functor from $\mathscr{L}$ to the **Sets**-based CCC.

All that isn't covered is the specific choice of sets to interpret the basic types, and elements of sets to represent the interpretations of the meaningful items. The latter is generally taken to be a parameter of variation for interpretation (perhaps subject to constraints), as is the choice of interpretation for the basic types other than P, which assumed standardly (always, as far as I know) fixed by the semantic theory. By the 'Functor Lemma' of appendix A (a trivial generalization of proposition 5.1 of LS), any choice of (c,d) in a CCC uniquely determines a functor from $\mathscr{L}$ into that CCC (and vice-versa). So the idea that an interpretation of a language is a CC functor from $\mathscr{L}$ encapsulates most of the fixed structure that has traditionally been attributed to the idea.

One consequence of this is that $\mathscr{L}$ provides us with a set of finitistic but variable-free entities for representing meanings, for which the typed lambda-calculus is a convenient but not necessary notation (where the variables can be eliminated thank to the Functional Completeness Theorem of LS, although the resulting formulations are tedious to produce and unpleasant to look at). This could be of interest for theories of 'structured meanings' where one would want to have finitistic representations of meaning that were as noncommittal as possible about details that are either completely random or motivated by whatever arcanities happen to be current in syntactic theory.[14]

[14]So that, for example, contrary to what appears to be an immediate consequence of

3. CCCs from Proof(-net)s

Our next step will be to get a CCC out of the proof-net, or, more generally, proof, which will contain a distinguished arrow, which a functor can then map into the model-theoretic semantics, which will factor through $\mathscr{L}$ and its interpretation functor. Note that this will be an utterance-specific CCC, rather than one that's a property of the whole language. We will call it the 'proof-CCC' $\mathscr{P}$.

The first step in constructing $\mathscr{P}$ is to notice that a correct proof-net can be seen not only as a valid sequent of multiplicative linear logic, but also as one of intuitionistic implication-conjunction logic ($\mathrm{NJ}_{\rightarrow,\wedge}$), which furthermore is 'balanced', that is, every proposition-letter (literal) that appears at all appears at most twice (this is also called the 'two-property'). This means that the following theorem proved in Babaev and Solov'ev (1982)[15] applies:

(10) B&S Theorem: A balanced sequent of $\mathrm{NJ}_{\rightarrow,\wedge}$ has at most one normal proof.

Since, with appropriate reinterpretation of the connectives, a balanced sequent of intuitionistic $-\!\circ\otimes$ logic is also one of $\mathrm{NJ}_{\rightarrow,\wedge}$, it follows that such a sequent will have a unique normal proof in this logic, which will be equivalent to an arrow in a CCC.

But to actually apply the theorem to our situation, a bit of joining and fitting is required, which is easier if we use the version of the theorem stated and proved in Troelstra and Schwichtenberg (2000, p. 274):

(11) TS.8.3.4: Let A, B be objects of a free CCC $\mathscr{C}$ over a set of proposition-letters not containing $\mathbf{1}$. If $[A \Rightarrow B]$ is balanced, then there is at most one arrow from A to B in $\mathscr{C}$.

To construe the proof-net as a balanced sequent, we can take the axiom-links as being distinct literals, and substitute them (by occurrences, not types) in the formulas of the glue sequent for the literals that they link. The

the position of King (2007), one would not need to work out a correct theory of the representation of case and agreement in Icelandic (a formidable problem) in order to investigate the semantics of belief reports in that language.

[15] With various later simplifications and generalizations, as discussed in Troelstra and Schwichtenberg (2000, ch. 8) and Aoto and Ono (1994).

resulting 'proof sequent' will obviously be valid, have the two-property, and have the original glue sequent as a substitution instance, under a substitution that we'll call τ. Construing the (syntactic) $\otimes$ and $\multimap$ connectives in the formulas as product and exponential of a CCC, with the axiom-links as basic objects, the formulas of the proof sequent are now objects of a free CCC; when we're talking about this, we'll designate the product and exponential of A, B as $A \times B$ and $[A \Rightarrow B]$, respectively. Next, for each (full, not sub-) formula of the proof-sequent, we adjoin to this CCC an arrow from $\mathbf{1}$ to the object/type constituted by that formula, so that we get one indeterminate for each occurrence of instantiated an instantiated meaning-constructor (a double negative, each of type $f_p \rightarrow f_p$, will produce two indeterminates). This produces the proof-CCC.

Note that we can also add indeterminates in the same way based on the formulas in the glue sequent, and thereby get a 'glue-CCC', but our uniqueness result won't apply, because this sequent lacks the two-property. Nevertheless, we will later find a use for this additional CCC.

Our claim is then:

(12) a. In the proof-CCC, there is an arrow from $\mathbf{1}$ to the final type (the formula to the left of the $\vdash$).

b. This arrow is unique.

(a) is quite straightforward, but I'll go through it anyway, while (b) requires a bit of modest fiddling. I'll look at these in turn.

For (a), we can construct the arrow by means of an easy adaptation of the ideas behind the algebraic correctness criterion of de Groote (1999), and Perrier's (1999) modification thereof. The method is to start by assigning arrows as labels to the premises of the proof-net, and apply rules to propagate these labels and others derived from them through the net, until a label gets assigned to the final conclusion (illustration to come). For deGroote, the label assignment is part of a method for proving correctness of a net, while for Perrier, it also provides 'semantic readings' (but only for implicational nets; tensors are not included). But for us, they will simply be a technique for constructing an arrow of the desired type.

The label-propagation is driven by the usual polarities:

(13) a. The formulas to the left of the $\vdash$ (premises) have negative polarity.

b. The formula(s) to the right of the $\vdash$ (conclusion(s)) have positive polarity (in intuitionistic logic, there is only one such formula).

c. The consequent of an implication has the same polarity as the whole implication.

d. The antecedent of an implication has the opposite polarity to the whole implication.

e. The components of a tensor have the same polarity as the whole tensor.

The meaning-labels will be called 'values', and propagate as follows, where the notation is the 'internal language' of the proof-CCC (LS:75), also discussed briefly in appendix A:

(14) a. The value of a premise is the lexical indeterminate associated with its (instantiated) meaning-constructor (different from the indeterminate of $\mathscr{L}$ associated with the lexical/uninstantiated version of the constructor).

b. The value of the negative antedecent of a positive implication is a new 'nonlexical' indeterminate of type A, where A is the type of the antecedent.

c. The value of the source of an axiom link is the value of its target.

d. If the value of the (positive) antecedent of a negative implication is $a{:}\mathbf{1} \to A$, and that of the entire implication is $f{:}\mathbf{1} \to [A \Rightarrow B]$, then the value of its consequent is $\mathbf{ev}^A_B{<}f, a{>}$, or just $f(a)$ for short.

e. If the value of the positive consequent of a negative implication is an arrow ϕ, and the value of the implication's positive antecedent is x_A, then the value of the entire implication is $\lambda_{x \in A}\phi$, as defined by the Functional Completeness Theorem of LS.

f. If the value of the left and right components of a positive tensor are ϕ, ψ, respectively, then the value of the entire tensor is $<\phi, \psi>$ (standard CCC pairing of arrows from a common source, in this case, $\mathbf{1}$).

g. If the value of a negative tensor is ϕ of type $A \times B$, then the values of its left and right components are $\pi^1_{A,B}(\phi)$ and $\pi^2_{A,B}(\phi)$, respectively.

Clause (f) isn't used in current glue analyses, due to the non-use of tensor-introduction, but is included for completeness, and in case somebody decides they want to use this rule after all. The calculation of values follows the flow of deGroote's algebraic correctness criterion, with the result that if the net is well-formed, the nonlexical indeterminates introduced at step (b) will all be eliminated in the value of the Final Conclusion. And the result will be an arrow from $\mathbf{1}$ to the Final Conclusion object/type/formula in the proof-CCC.

Now we use the B&S theorem to show that this arrow is unique. The obstacle is that the theorem applies to arrows in the free CCC over the axiom-links, without any indeterminates, whereas what we have is an arrow with them, in the proof-CCC. This can be fixed by using Functional Completeness, which allows us to assert the following:

(15) If $c{:}\mathbf{1} \to C$ is an arrow in $\mathscr{C}[x_1]\ldots[x_n]$, $\mathscr{C}$ a CCC, $x_i{:}\mathbf{1} \to A_i$ for $i = 1, \ldots, n$, then there is a unique $c_F{:}A_1 \times \ldots \times A_n \to C$ such that $c_F<x_1, \ldots, x_n> = c$ (associations to the right, for definiteness).

Now suppose we have a potentially different $c'{:}\mathbf{1} \to C$ in $\mathscr{C}[x_1]\ldots[x_n]$. We can eliminate the indeterminates to restate c' as $c'_F<x_1, \ldots, x_n>$ But c'_F is from $A_1 \times \ldots \times A_n$ to C, and is balanced, so, by BS' theorem, $c'_F = c_F$, therefore $c = c'$.

The result we claim to have proved can be stated as follows:

(16) Suppose $x_i{:}\mathbf{1} \to A_i$ for $i = 1, \ldots, n$ are indeterminates adjoined to a free CCC $\mathscr{C}$ such that $A_1, \ldots, A_n \vdash C$ is valid and balanced. Then there is one and only one element of C in $\mathscr{C}[x_1]\ldots[x_n]$.

This allows us, in a 'philosophically' natural way (as opposed to any specific technical sense of 'natural') to rapidly escape from the unfamiliar and mathematically somewhat treacherous world of SMCCs that linear logic proofs

inhabit, into the more familiar realm of CCCs, where much of formal semantics has been traditionally conducted.

But we've only shown this for proof-nets; what about the other, more often used, glue proof formats? For Gentzen sequents, there's clearly no problem at all, since their proofs start with instances of Axiom, which induce the required pairing. And linear universal quantification[16] won't be a problem, since it is introduced further down in the proofs, and so doesn't affect the pairing.

Natural deduction can also be managed; here we work through 'tree style'. Universal quantification is more of a problem here, but can be managed by shifting the quantifiers to the top levels of the meaning-constructors, and eliminating them first, leaving a purely propositional deduction that can be handled as follows.

To set things up, we first define the 'frontier' of an ND (tree-style) deduction:

(17) Frontier: The frontier of an ND tree-style deduction consists of the formulas constituting the undischarged premises and the conclusion.

There's also a proviso:

(18) For a zero-step (one formula) deduction, the formula is counted twice, once as premise, and once as conclusion

Now we define the pairing, inductively, as follows. Note that the pairing relation is not considered to have a direction:

(19) a. For a zero-step (one formula) deduction, each literal is paired with itself, in once instance as a premise literal, in the other as a conclusion literal.

b. For an implication elimination step:

$$\dfrac{\begin{array}{cc}\Gamma & \Delta \\ \vdots & \vdots \\ \alpha & \beta \\ A & A\multimap B\end{array}}{B}\ \multimap\text{-elim}$$

[16] In either a restricted version of Girard's system F, or a first order approach, as discussed in Kokkonidis (2008). What quantification is supposed to do is allow certain nodes in the glue structure to associate with anything in the f-structure, but the results can be achieved by other means, as discussed in Andrews (2008).

Add a link from each literal in the conclusion of α to the corresponding one in the antecedent of the conclusion of β (both A), and one from each literal in the consequent of the conclusion of β to its correspondent in the new conclusion produced by the rule (both B).

c. For an implication introduction step:

$$\dfrac{\begin{array}{c}\Gamma, [A]^i \\ \vdots \alpha \\ B\end{array}}{A \multimap B} \multimap\text{-intr}^i$$

Add a link from each literal in the premise to be discharged of α to the corresponding one in the antecedent of the new conclusion of the rule (A), and one from each literal in the conclusion of α to the consequent of the new conclusion of the rule (B).

d. For a tensor elimination step:

$$\dfrac{\begin{array}{cc}\Gamma & [A]^i, [B]^j, \Gamma \\ \vdots \alpha & \vdots \beta \\ A \otimes B & C\end{array}}{C} \otimes\text{-elim}^{i,j}$$

Add links from the literals A, B in the conclusion of α to those of the corresponding premises being discharged of β, and from the conclusion C of β to the new conclusion being produced by the rule.

Tensor introduction is obvious, and not currently needed for glue.

The following are then straightforward by induction:

(20) a. Each literal on the frontier is connected to something by only one link (possibly itself, if the Proviso applies).

b. From each literal on the frontier there is a unique nonrepeating path to a literal on the frontier, which will be a different literal unless the Proviso applies.

c. The paths partition the set of literals in the proof (both on and not on the frontier).

d. Assigning a distinct new proposition-letter to each equivalence class, and replacing each literal in that class with this new letter produces a valid proof, that has the original as a substitution instance.

e. This new proof proves a sequent with the two-property.

We now have a propositional glue proof to which the procedure of (13) can be applied.

Glue proofs thus give rise in a natural manner to CCCs, which can then be interpreted by means of functors into **Sets**.[17] But there turns out to be substantively more than one way to do this.

4. Interpreting Sentences

Given the results of the previous section, a glue proof produces a proof-CCC $\mathscr{P}$ with a 'final type' (the type of the final conclusion of the proof) that has one and only one element. To interpret the sentence model-theoretically, we will want a functor from $\mathscr{P}$ into the semantics category $\mathscr{S}$. And, in order to produce the same results as the traditional formulations, this functor can't be chosen arbitrarily, but must be determined by the interpretation functor $I_{\mathscr{L}}$ for the lexical category $\mathscr{L}$, together with the way in which the $\mathscr{P}$ has been constructed by the syntax and the glue.

The indeterminates of $\mathscr{P}$ represent particular occurrences of meanings residing in words or formatives (any given word or formative can introduce multiple meaning-constructors), and it is assumed that all such occurrences are interpreted in the same way.[18] That is, all indeterminates of $\mathscr{P}$ that have been adjoined because of an occurrence of an instantiated version of a given lexical meaning constructor must get the same interpretation, which will be the same as the interpretation of that (uninstantiated) constructor under $I_{\mathscr{L}}$. As a consequence, a functor interpreting P in $\mathscr{S}$ can be factored through $I_{\mathscr{L}}$.

[17] As noted in the discussion at the bottom of LS:80.

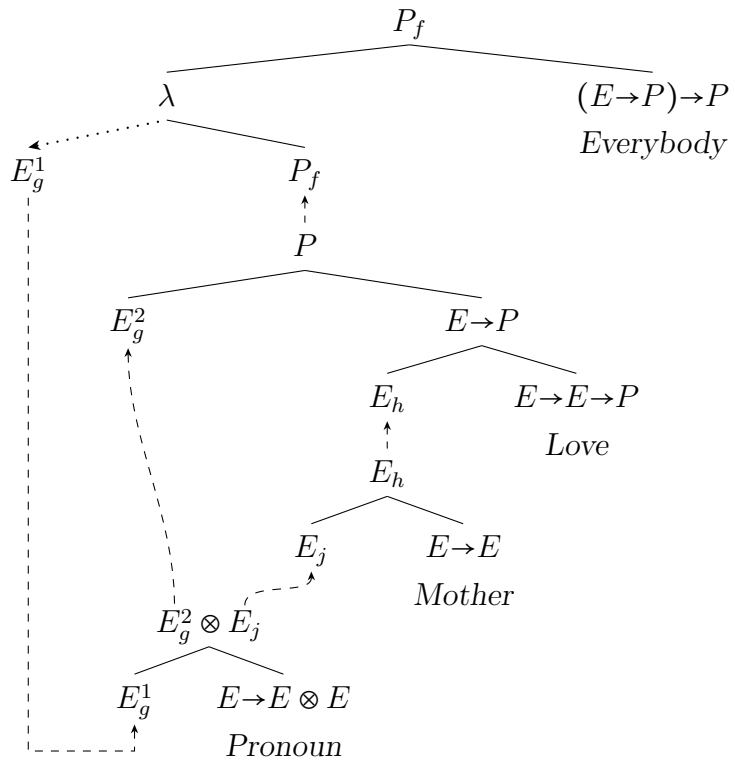
[18] A potential kind of counterexample to this would be a spoken sentence such as "I've tried to explain this to you, but you don't seem to understand, so now Fred here will EXPLAIN it to you", which suggests that Fred's version of explanation will be substantially more robust than mine. I would suggest that the emphatic stress on the second instance of *explain* is a way of temporarily spawning a second, nonstandard, lexical meaning, for use in this particular utterance.

We can see this by using the functor lemma again. We can map each indeterminate of $\mathscr{P}$ onto the one of $\mathscr{L}$ that it is an instantiation of, and each basic type of $\mathscr{P}$ onto the semantic type of the pair of glue literals that constitute it, and it follows immediately that this determines a CC functor of $\mathscr{P}$ to $\mathscr{L}$. Composing this with $I_{\mathscr{L}}$ will furthermore map the indeterminates and basic types of $\mathcal{P}$ onto the same values in $\mathscr{L}$ as the direct method, and so produce the same functor.

The fact that, under standard assumptions, we can factor the interpretation of $\mathscr{P}$ through that of $\mathscr{L}$ suggests that it might make no difference whether we do or not. But the two approaches turn out not to be equivalent, in that the indirect method turns out to provide some options that the direct method doesn't. This can be seen by examining the tensor-based analysis of pronouns, as presented for example in Asudeh (2004) (and earlier work). Consider the proof-net for a sentence such as *everybody loves their mother*, which is given here in the nonstandard format discussed in Andrews (2008):[19]

[19]The difference is that the links other than those from a positive implication to its negative antecedent are replaced by the dynamic graph of de Groote (1999), and, notationally, the former are written with dotted lines, and axiom links with dashed ones. The motivation is to make the nets look more like standard formats for linguistic representation. It is also more convenient for writing in labels for the nodes, since the longer labels tend to appear higher up in the display, where there is more space available.

(21)



Here the positive implication node has been labeled with a λ, in order to highlight the point that that's really what these nodes do (since the type is obvious from the daughters, leaving it out of the diagram does not impair intelligibility). More significantly, we've added superscripts to the E_g nodes, because in the proof-CCC they correspond to two objects, one for each axiom link. Doing this tends to fend off some easy-to-make errors, which I have made more than once.

So now, if the (negative) antecedent of the λ is assigned the indeterminate $x^{E_g^1}$, then the value that eventually comes back to its consequent (right daughter) will be something that can be written like this:[20]

(22) $\lambda_{x\in E_g^1} Love(Mother(\pi^1_{E_g^1,E_h} Pronoun(x)))(\pi^2_{E_g^1,E_h} Pronoun(x))$

In the direct approach, we simply have to stipulate that *Pronoun* is to be interpreted as $<\mathbf{Id}_{I_{\mathscr{L}}(E)}, \mathbf{Id}_{I_{\mathscr{L}}(E)}>$.

[20]Note that in a finicky notation, multiple occurrences of any given semantic constant should be differentiated by superscripts, because they represent distinct indeterminates of $\mathscr{P}$, although this information would also be implicit in the types of the indeterminates: in $\mathscr{P}$, there can't be two indeterminates of the same type.

But with indirect interpretation via $\mathscr{L}$, there are two possibilities. The first is to mimic the direct approach and map the *Pronoun* indeterminate of $\mathscr{P}$ onto a *Pronoun* indeterminate of $\mathscr{L}$, which gets interpreted in $\mathscr{L}$ as $\langle \mathbf{Id}_{I_{\mathscr{L}}(E)}, \mathbf{Id}_{I_{\mathscr{L}}(E)} \rangle$. The other is to map *Pronoun* onto $\langle \mathbf{Id}_E, \mathbf{Id}_E \rangle$, a.k.a. $\lambda_{x \in E} \langle x, x \rangle$, in $\mathscr{L}$ itself. This is what people are doing implicitly, I think when they write something like (23a) rather than (23b) as the meaning-constructor for anaphoric pronouns:

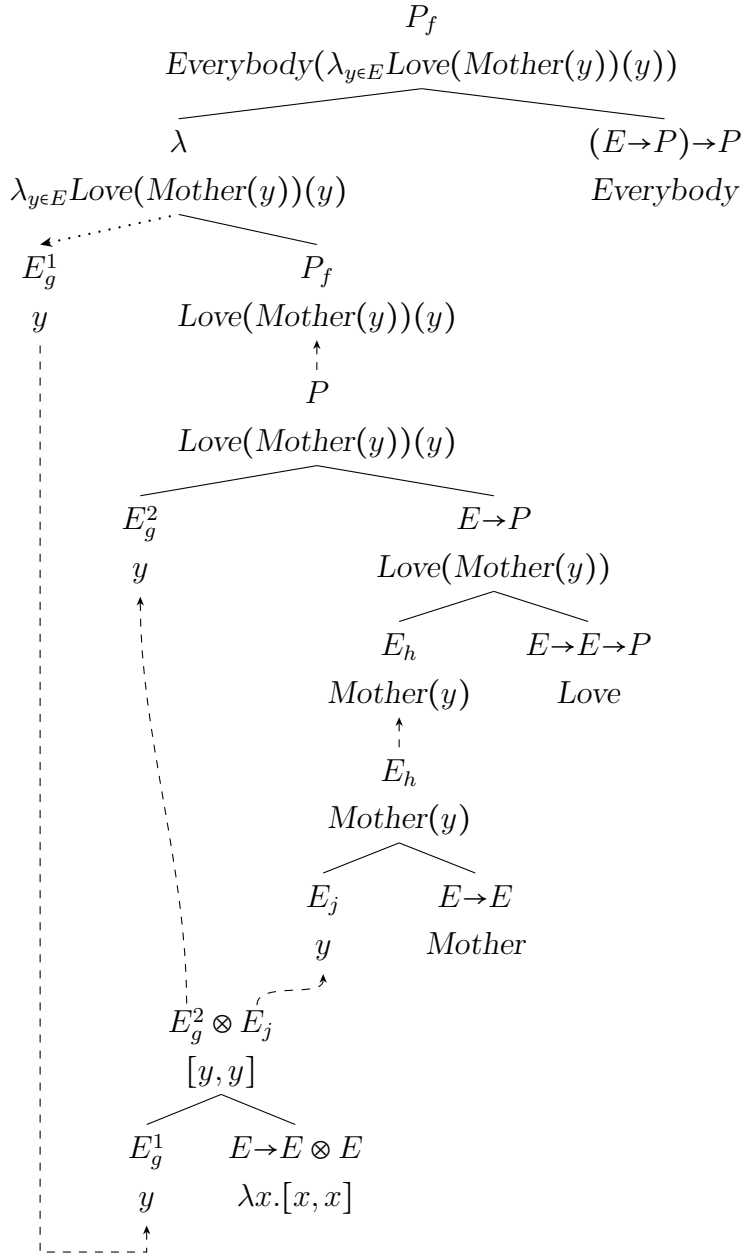
(23) a. $\lambda x.[x, x] \ : \ (\uparrow \mathrm{ANT})_e \multimap (\uparrow \mathrm{ANT})_e \otimes \uparrow_e$

b. *Pronoun* $: \ (\uparrow \mathrm{ANT})_e \multimap (\uparrow \mathrm{ANT})_e \otimes \uparrow_e$

But there is a considerably lack of clarity as to exactly what it is supposed to mean to write one thing rather than another on the meaning-side of a meaning-constructor, beyond just putting a constraint on the interpretation, especially in the absence of any overall theory of what kinds of constraints, if any, are supposed to apply to the meaning-side.

A pleasant notational feature of the indirect approach is that it renders it sensible to label the nodes of a proof-net with the images in $\mathscr{L}$ of the labels computed in accordance with (14), as for example illustrated below:

(24)



In particular, we can do β-reductions in $\mathscr{L}$ as we go, so that we there is no reason to ever look at the label (22), instead of the much more attractive:

(25) $\lambda_{x\in E}Love(Mother(x))(x)$

Note that although this labeling looks rather like Perrier's 1999 'maximal labeling', extended to include tensors, it's really a different kind of thing, since not in the category associated with the glue proof itself, but in a functorial image thereof. Therefore we should not expect to be able to get a

version of the Correctness Criterion out of it. It can be regarded as a finitistic representation of the meaning that exists automatically, has all of the mathematical structure of the intended model-theoretic interpretation that is relevant for the syntax-semantics interface, and has no arbitrary properties that the model-theoretic interpretation doesn't have as well).

Annotations in $\mathscr{L}$ can also be used in conventional deductive formulations of glue, in place of the linear **let**-terms that are normally used for the rule of $\otimes$-elimination, which can therefore be written as (a) below rather than (b), omitting the type information from the labels, as is customary:

(26) a.
$$\dfrac{\begin{array}{cc} w : A \otimes B & \begin{array}{c}[x : A]^i [y : B]^j \\ \vdots \\ z : C\end{array}\end{array}}{[\pi^1(w)/x, \pi^2(w)/y]z : C}\ \otimes\text{-elim}^{i,j}$$

b.
$$\dfrac{\begin{array}{cc} w : A \otimes B & \begin{array}{c}[x : A]^i [y : B]^j \\ \vdots \\ z : C\end{array}\end{array}}{\textbf{let}\ w\ \textbf{be}\ x \times y\ \textbf{in}\ z : C}\ \otimes\text{-elim}^{i,j}$$

Labels in the format of (a) could be regarded as inherently better than the standard one oF (b), since the latter merely restates the structure of the proof, without even registering the role of commutative conversions with $\otimes$-elimination,[21] while those such as (a) can be formulated to carry the information that the proofs are being interpreted in a CCC.

But are these proof terms merely convenient and relatively attractive to look at, or do they also have some empirical teeth?

One possible source of empirical evidence might be phenomena such as the Weak Crossover constraint. But in LFG there is already an ANTECEDENT grammatical function involved in the analysis of pronouns, which might be invoked for this purpose, so probably not.

Another emerges from considering what the goals of formal semantics might be taken to be. The conventional story as told at the beginning of formal semantics textbooks is that the semantic theory is:

[21] Proofs that wind up having the same projection-based proof-terms will be related by the standard commuting conversions.

(27) a. Saying something about the relationship between language and the world.

b. Characterizing entailment and other semantically based properties and relationships between sentences (such as 'appropriate answer to a question').

(b) is probably not something that many people in contemporary linguistics would take issue with, at least as an aspiration, while (a) is more problematic: in a nutshell, Realists think it is obviously essential in order to have something that can reasonably be called 'semantics', while Anti-Realists think it makes no sense whatsoever. Regardless of what one thinks about this, it is plausible to set up finitistic representations that can help with (b), at least for immediately perceptible entailments, of the kind that form almost all of the data of formal semantics. For this purpose, the format of (25) would seem to be more likely to be useful than that of (22).

Another possible source for an empirical argument might be found in Asudeh's (2004, 2008) analysis of 'resumptive pronouns', and Asudeh and Toivonen's (2009a) analysis of 'copy raising'. These work by postulating a 'resource manager' meaning-constructor, which, in effect, gobbles up an excess pronoun's meaning and throws it away.

A typical form for such a constructor is M in (28) below, where p is the f-structure correspondent of a pronoun, a that of its antecedent:

(28) Pronoun : $\lambda x.[x, x]$: $E_a \multimap E_a \otimes E_p$
M : $\lambda P x.x$: $(E_a \multimap E_a \otimes E_p) \multimap E_a \multimap E_a$

The first argument of M consumes the entire content of the pronoun, so that the net result is just the identity function on the antecedent, and the structure is consequently interpreted as if the pronoun wasn't there.

This creates the problem of seeming to allow that there might be constructors that throw away meanings with more content than pronouns, such as for example a transitive verb *galk* that simply ignores the meaning of its object, and attributes the property of 'walking' to its subject:

(29) a. John galked Mary ('John walked')

b. John galked everybody ('John walked')

Such a constructor would be easy to write, with a meaning-side similar to that of (28), but should presumably be somehow forbidden:

(30) $\lambda yx.Walk(x) \ : \ (\uparrow \text{OBJ})_E \multimap (\uparrow \text{SUBJ})_E \multimap \uparrow_P$

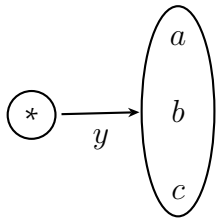
So what might be the difference between the putatively occurring (28b) and the presumably impossible (30)?

One difference is under indirect interpretation, (28b) is only throwing away combinators of $\mathscr{L}$, while (30) would have to toss lexical indeterminates. It is reasonable to suppose that this might be impossible, although I don't know at this point what kind of general mathematical idea such a constraint would be an instance of. But interpretation through $\mathscr{L}$ provides a relevant distinction, while direct interpretation does not.

I'll close this section by saying something about the status of nonlexical indeterminates such as y in (24). These look like variables, and in a sense, that's what they are, suggesting that a semantic label such as $Love(Mother(y))(y)$ requires some kind of assignment function to be interpreted model-theoretically. But this is not the case: y can simply be treated as a non-lexical indeterminate, adjoined to $\mathscr{L}$, and for that matter to $\mathscr{S}$, if we want to have a full model-theoretic interpretation of the expression 'before' λ-binding.

Since the possibility of interpreting variables model-theoretically without assignments by adjoining indeterminates to **Sets** doesn't appear to be widely known in NL semantics, it might be worth illustrating it with a picture, where we have a model with a three element universe, plus an indeterminate y adjoined:

(31) **1** E



The indeterminate can be thought of as a kind of 'blob-connector', which relates sets 'from the outside', without having any content w.r.t. their actual

members/elements. I conjecture that this idea might be useful for dealing with certain phenomena in opaque contexts, such as reference to nonexistent entities such as Gandalf of Middle-Earth, or Mabel's impossible beliefs about having arthritis in her thigh.

5. Yet another CCC

In this section, I will make an empirical case for the usefulness of yet another CCC, standing between the proof-CCC and the lexical CCC, which I will call the 'glue CCC' $\mathscr{G}$, because its basic objects are the literals of the glue-proof (semantic type and f-structure information, but not the pairing), rather than the axiom-links. We can construct it by taking the free CCC with these literals as basic objects, and adjoining one indeterminate for each instantiated constructor. These will be in the obvious one-to-one correspondence with the indeterminates of $\mathscr{P}$, and we can produce a functor $P\colon\mathscr{P}\to\mathscr{G}$ by using this correspondence to map the indeterminates, and using τ from the previous section to map the basic types of $\mathscr{P}$ onto the literals (as types) that they were produced from by axiom-pairing. It is furthermore obvious that any functor $L\colon\mathscr{P}\to\mathscr{L}$ will factor through P in a unique way.

$\mathscr{G}$ and P exist 'for free' whether we want to do anything with them or not, but do they do anything useful? A possibility is that they might provide better solutions to some problems that the ones presented in Andrews (2007) and Andrews (2010).

The first problem is to explain the evidence given by Alsina (1996) that the 'Causee Agent' in sentences like (32) below is not a Subject, in contrast with (33), where the 'Persuadee Agent' appears to be one, the evidence being that the 'floating quantifier' *cadascun* can refer to the Persuadee Agent in (33), but not the Causee Agent in (32):

(32) Els metges ens deixen beure una cervesa cadascun
the doctors us let drink a beer each

a. Each of the doctors lets us drink a beer

b. *The doctors let each of us drink a beer

Alsina (1996, p. 217).

(33) Els metges$_i$ ens$_j$ han convençut de beure una cervesa cadascun$_{i/j}$
the doctors us have convinced of drink a beer each

a. The doctors each convinced us to drink a beer

b. The doctors convinced us to drink a beer each

Alsina (p.c.), reported in Andrews (2007)

According to the theory proposed in that paper, the meaning-constructors for the causative and caused verbs in (32) would be:

(34) $\lambda Pyx.Cause(P(y))(y)(x)$:
$((\uparrow \mathrm{OBJ}_{Rec})_e \multimap \uparrow_p)) \multimap (\uparrow \mathrm{OBJ}_{Rec})_e \multimap (\uparrow \mathrm{SUBJ}) \multimap \uparrow_e$
$Drink$: $(\uparrow \mathrm{OBJ})_e \multimap (\uparrow \mathrm{OBJ}_{Rec})_e \multimap \uparrow_e$

The problem is that there turns out to actually be no real reason why the middle argument of 'Cause' and the last argument of 'Drink' shouldn't be associated with the SUBJ grammatical function rather than OBJ_{Rec} (object associated with the Recipient semantic role, traditionally 'indirect object'). Andrews proposes a rather clumsy 'Functional Consistency' constraint to deal with this; here I will suggest an alternative using the glue-CCC.

The proposal is that most meaning-constructors, that is, all except for a limited, universally specified few, must saturate any higher order arguments they have in the glue-CCC. To see how this works, consider these alternative constructors, which, counterintuitively, work under standard glue:

(35) $\lambda Pyx.Cause(P(y))(y)(x)$:
$((\uparrow \mathrm{SUBJ})_e \multimap \uparrow_p)) \multimap (\uparrow \mathrm{OBJ}_{Rec})_e \multimap (\uparrow \mathrm{SUBJ}) \multimap \uparrow_e$
$Drink$: $(\uparrow \mathrm{OBJ})_e \multimap (\uparrow \mathrm{SUBJ})_e \multimap \uparrow_e$

Given the proposed constraint, these constructors are impossible, since, if the subject and 'upper object/Causee Agent' of the causative f-structure f are g, h, respectively, the types of the three arguments will be $g_e \rightarrow f_p$, g_e and h_e, respectively, and it won't be possible to saturate the property argument and still get the right meaning (object of Cause is performing the caused action, but we can't apply a function of type $g_e \rightarrow f_p$ to an argument of type h_e).

But with the correct meaning-constructors of (34), the meaning-side after instantiation will be:

(36) $\lambda P^{h_e \to f_p} y^{h_e} x^{g_e}.Cause(P(y))(y)(x)$

This satisfies the constraint, and works.

Aside from being arguably simpler and more elegant than the Functional Consistency constraint, the new constraint also deals with the problem considered in Andrews (2010), of blocking the possibility of adjectives such as the hypothetical *alleger*, with a meaning such that *an alleger murderer* is somebody who has alleged that somebody (else) is a murderer. It is quite easy to write a constructor for such an adjective under the standard assumption that 'modal' adjectives are of the semantic type $(e \to p) \to e \to p$, but the proposed constraint blocks this, without creating a problem for occurring modal adjectives such as *alleged*, *former*, and *self-proclaimed*.

The way these adjectives work is that if an NP's f-structure is g, the noun's constructor will be of the form $NMeaning : g_e \multimap g_p$, so that a modal adjective such as *alleged* can have the (conventional) constructor:

(37) $\lambda P x.(\exists y)(Allege(P(x))(y)) \; : \; (g_e \multimap g_p) \multimap g_e \multimap g_p$

The undesired *alleger* constructor can be obtained just by swapping the roles of x and y in the arguments of *Allege*, but the proposed constraint will block this move, since P will be of type $g_e \to f_p$, requiring to be satisfied by a variable of type g_e. x in (37) will be of this type, but we can assume that y cannot be, due to being internally produced and bound by a constructor-internal existential quantifier, thereby forbidding the role-swap needed to produce **alleger*. This is a considerably smoother account than the one provided by Andrews.

It would be nice if we could proclaim that all higher order arguments needed to be saturated in the glue-category, but this would appear to block various necessary things such as the quantificational determiners *all* and *most* (assuming a generalized quantifier format). We therefore retain Andrews' proposal that there is a difference between 'lexical' meaning-constructors that form an open class, subject to various constraints, and a limited class of 'grammatical' meaning-constructors that are specified by UG, and therefore not subject to the constraints that apply to the lexical ones.

The causative problem could reasonably said to be closely tied to the architecture of LFG plus glue semantics, and therefore of limited interest,

but the nonexistence of adjectives like **alleger* is a genuine problem for many different accounts of formal semantics, both unnoticed and unsolved prior to Andrews (2010). I suggest that its nature and solution is closely related to some of the underlying intuitions of Generative Semantics and the Minimalist Program applied to Government- Binding Theory,[22] the GS/MP idea being that the basic types 'project into the syntax' (so that, for example 'VP-complement' structures must have 'underlying subjects' in some sense or another (Andrews (1976), Boeckx et al. (2010), and a great deal of intervening literature), and UG then imposes limits of some kind on what can be done with them (central to MP, somewhere between absent and poorly developed in 1960s GS).

6. Possibly Even Wrong

So we see that the idea that variable-like entities require assignments for interpretation is not in general true, but, on the other hand, this idea, and the entire functorial approach to semantic interpretation for glue developed here, works only under certain circumstances, where something like the Functional Completeness Theorem of LS holds. This is clearly not the case for certain kinds of systems that have been set up to model (hyper)intensional phenomena, such as the most general ones of Muskens (2007), because, in them, alpha-equivalence over lambda-expressions doesn't hold. It does in the system he actually proposes for English semantics later in the paper, but this system is not explicitly set up as a CCC, so there are further opportunities for functorial interpretation not to work out. On the other hand, the hyperintensional semantics of Pollard (2008) seems explicitly set up so as to provide a CCC to interpret into, so it would be surprising if a big problem appeared there. And of course there are many other approaches to semantics, such as situation semantics, which aren't obviously in the CCC format (**Sets**-based or otherwise), giving further possibilities for empirical failure.

It therefore seems to be the case that the general approach given here manages to clamber over the 'not even wrong' hurdle, which is interesting

[22] Taking Chomsky at his word to the effect that the basic idea of the Minimalist Program is to conduct a conceptually-driven garage-cleaning operation, so that the current MP is what you get by doing this with GB as a starting point.

even if it turns out to be a wrong theory rather than a right one.

But there is a less 'positive' problem, which is that of dealing with dynamic phenomena such as donkey sentences, presupposition, intersentential anaphora, etc. Here the CCC-based approach is simply not doing some things that seem to need to be done, rather than doing something in a possibly wrong manner. There are of course by now several approaches to doing some kind of dynamic semantics in glue,[23] but they all involve what appear to me to be essentially ad-hoc extensions of the glue notation in order to capture specific empirical effects, without emerging from more basic ideas in a principled way. So it remains to be seen whether the CCC functor idea can be extended to deal with dynamic phenomena in a 'philosophically natural' way.

Appendix A Some CCC Basics

A.1. Categories

In this appendix I will work through the basic category theory used in the text. The discussion will be too brief to serve as a useable introduction, and too basic and shallow to be useful to people who know the subject, but might be of some utility to some people in various intermediate states.

A category consists of a collection of 'objects' and a collection of 'arrows', obeying some laws:

(38) a. Every arrow has a unique object as its 'source', and another unique object as its 'target'. An arrow f with source A and target B is notated $f{:}A \to B$ (same as standard notation for a function f from A to B, for good reason, as we shall see).

b. Given $f{:}A \to B$ and $g{:}B \to C$, there is $gf{:}A \to C$ (Composition; sometimes '$\circ$' is used as a symbol for composition, but not here).

c. Given $f{:}A \to B, g{:}B \to C, h{:}C \to D$, $(hg)f = h(gf)$ (Associativity).

d. For each object A, there is $\mathbf{Id}_A{:}A \to A$ such that for any $f{:}A \to B$, $f\mathbf{Id}_A = f = \mathbf{Id}_B f$ (Identities).

[23] The ones I know about being Crouch and van Genabith (1999), Kokkonidis (2005) and Lev (2007).

A very important example is the category **Sets**, with sets as objects and slightly modified functions as arrows. The modification is that the functions must be packaged with some specific set containing their range, to serve as their target. It is obvious that with this modification, the laws of (38) are satisfied.

The other example that will be important for us is (certain kinds of) deductive systems. Here the objects are formulas, and the arrows proofs. Composition consists of chaining a proof f of B from A with a proof of g of C from B (to get a proof gf of C from A, corresponding to 'cut' in proof-theoretical terminology), while Associativity represents the fact that if we do this multiple times, we care only about the order of the sub-proofs, not their grouping. Finally, the identities consist of proving a formula from itself by doing nothing (the identity axiom, in many proof systems); appending this step to either end of a proof is construed as not changing it in any way.

This account applies only to certain kinds of proofs, because there is an annoying restriction that the proofs-as-arrows can only derive a single conclusion from a single premise; to go beyond this, one needs to result to 'multicategories', which seem to introduce a lot of complexity. But it seems to (almost?) always be possible to work around this problem in one way or another.

A number of basic concepts and proofs of elementary abstract algebra have correspondences in category theory. So we can prove that identities are unique, by the same technique as applies to monoids and groups:

(39) Suppose $\mathbf{Id}_A$ and 1_A are both identities for A. Then $\mathbf{Id}_A = 1_A\mathbf{Id}_A = 1_A$.

Inverses can be defined as follows:

(40) $g{:}B \to A$ is an inverse of $f{:}A \to B$ iff $gf = \mathbf{Id}_A$ and $fg = \mathbf{Id}_B$ (basically the same as in the 'algebra of functions').

Inverses don't have to exist, but if they do, they are unique:

(41) Suppose g, h are inverses of f. Then $g = g(fh) = (gf)h = h$.

An arrow with an inverse is called an 'isomorphism', and two objects connected by such an arrow are called 'isomorphic'. It follows that in **Sets**, all sets of the same cardinality are isomorphic.

Groups, monoids, etc, also form categories, with the specific systems as objects, and homomorphisms between them as arrows; you can see that the concept of isomorphism then is equivalent to the standard algebraic one. A final subtlety is that the 'collections' of objects and arrows don't have to be sets, they can be proper classes (as in the category of groups, or **Sets** itself), or in principle even bigger things. Categorists appear to consider it to be uncool to indulge in 'size worries', under most circumstances.

A.2. Cartesian Closed Categories (CCCs)

CCCs obey various additional laws beyond those for mere categories. Intriguingly for linguists, these appear to be exactly what is needed for the syntax-semantics interface part of 'Montague Grammar'.

For a 'Cartesian Category', we require first some objects:

(42) a. A 'terminal object' $\mathbf{1}$.

b. For each pair of objects A, B, a product object $A \times B$.

And some arrows:

(43) a. For each object A, a 'terminal' arrow $\bigcirc : A \to \mathbf{1}$.

b. For each pair of objects A, B, the 'projection arrows' $\pi^1_{A,B} : A \times B \to A$ and $\pi^2_{A,B} : A \times B \to B$.

c. For each pair of arrows $f : C \to A, f : C \to B$, a 'pairing arrow' $<f, g> : C \to A \times B$.

This stuff also obeys some equations:

(44) a. For any $f : A \to \mathbf{1}$, $f = \bigcirc_A$.

b. For any $f : C \to A$ and $g : C \to B$, $\pi^1_{A,B}<f, g> = f$ and $\pi^2_{A,B}<f, g> = g$.

c. For any $f : C \to A$, $g : C \to B$ and $h : D \to C$, $<f, g>h = <fh, gh>$.

It is evident that **Sets** constitutes a CCC, with any one element set chosen as a terminal object, standard cartesian products as the product, and the obvious functions as projections and pairing.

This example highlights the important fact that there is often a considerable element of arbitrariness in the choice of the objects for constitution of a CC, for example any one element set will do as the terminal object. $\{*\}$ is the standard notational choice. But these are all isomorphic, so, to the categorist, which is chosen doesn't matter. Another important point is that an arrow from the (chosen) terminal object to a set functions equivalently to a conventional member of the set (whichever member the arrow assigns as value to the sole member of the terminal object). Such arrows are called 'elements', and are essential to some of the techniques that category theory enables, such as constructing model theoretic interpretations of typed lambda calculus expressions without using assignments of values to variables.

In the deductive world, the terminal object is an 'always true' proposition $\top$ that can be concluded from any proposition, while the product is conjunction, usually symbolized as $\wedge$. The projections are the two cases of the standard rule of $\wedge$-elimination:

(45) $A \wedge B \xrightarrow{\pi^1_{A,B}} A \qquad\qquad A \wedge B \xrightarrow{\pi^2_{A,B}} B$

Pairing is a step that combines proofs f of A from C and proofs g of B from C into a single proof of $A \wedge B$ from C:

(46) $$\frac{C \xrightarrow{f} A \qquad C \xrightarrow{g} B}{C \xrightarrow{< f, g >} A \wedge B}$$

This is different in formulation to the standard conjunction introduction rule, but has the same ultimate effects in the single-premise environment. We are also not using the bar in the manner of tree-style natural deduction proofs, but of sequent (both Gentzen and natural deduction) proofs, where it can be read as saying 'if the proof(s) above the bar exist, then so does the one below it'.

Finally, the equations correspond to the standard rules of proof-normalization, as discussed in sources such as Girard et al. (1989), whereby various trivial variants of proofs are identified by eliminating pointless detours.

To push on to a CCC, we need the following additional items:

(47) a. For each pair of objects A, B, an 'exponential object' $[A \Rightarrow B]$ (there are various notations for this).

b. For each pair of objects A, B, an 'evaluation arrow' $\mathbf{ev}_B^A{:}[A \Rightarrow B] \times A \to C$.

c. For every arrow $h{:}C \times A \to B$, a 'curried arrow' $\mathbf{cur}(h){:}C \to [A \Rightarrow B]$.

These equations must be obeyed, which are more intimidating that what we have seen previously:

(48) a. For $h{:}C \times A \to B$, $\mathbf{ev}_B^A{<}\mathbf{cur}(h)\pi^1_{C,A}, \pi^2_{C,A}{>}$.

b. For $k{:}C \to [A \Rightarrow B]$, $\mathbf{cur}(\mathbf{ev}_B^A{<}k\pi^1_{C,A}, \pi^2_{C,A}{>})$.

They enable functions of multiple arguments and their curried versions to be essentially equivalent.

In **Sets**, $[A \Rightarrow B]$ is the standard function space B^A, while $\mathbf{ev}_B^A$ is the 'evaluation function' that, given an member f of B^A and a of A, applies f to a producing the member $f(a)$ of B. $\mathbf{cur}(h)$ on the hand is the curried version of a function on two arguments that converts it to a function that takes the first argument to a function that takes the second argument to the final result.

In proofs, exponential objects are implicational formulas, while $\mathbf{ev}_B^A$ is the proof-step of implication elimination (modus ponens), and $\mathbf{cur}(h)$ is implication introduction (hypothetical deduction) applied to the proof h:

(49)
$$[A \Rightarrow B] \wedge A \xrightarrow{\mathbf{ev}_B^A} B \qquad \frac{C \wedge A \xrightarrow{h} B}{C \xrightarrow{\mathbf{cur}(h)} [A \Rightarrow B]}$$

A test of calculational facility is to work out that the two operations below are inverses:

(50) a. 'name of': for any $f{:}A \to B$, $\ulcorner f \urcorner{:}\mathbf{1} \to [A \Rightarrow B] =_{def} \mathbf{cur}(f\pi^2_{\mathbf{1},A})$

b. 'func of': for any $k{:}\mathbf{1} \to A \Rightarrow B$, $k^f{:}A \to B =_{def} \mathbf{ev}_B^A{<}k \bigcirc_A, \mathbf{Id}_A{>}$

'name of' lets you convert an intuitive function-as-arrow such as $\mathit{Sleep}{:}E \to P$ (a function from entities to propositions) to an element of the function-space $[A \Rightarrow B]$ (an arrow from $\mathbf{1}$), from which 'func of' lets you get the original arrow back. This is important for the technique for interpreting lambda expressions without assignments.

The deductive system as developed so far is a 'monosequent' version of intuitionistic conjunction-implication logic, typically notated as something along the lines of $\mathrm{NJ}_{\rightarrow,\wedge}$. It can be seen as the non-linear counterpart to glue logic, derivable from it in the Gentzen sequent formulation by adding the 'structural' rules of Weakening and Contraction, and by somewhat less transparent means in other formulations of logic.

Any proof in the monosequent formulation can be recast in conventional 'multisequent' natural deduction, and vice-versa, by first taking the initial step of packing all of the premise formulas into a conjunction. But the fact that this step is not in general reversible in a natural way (if one of the premises is a conjunction, how do you know whether to split it or not in unpacking?) means that the two kinds of formulations aren't quite equivalent. But this is a problem that tends to be easier to work around (for example as in Troekstra and Schwichtenberg's reformulation of BS' theorm) than through (by developing 'multicategories').

A.3. (CC) Functors

The final idea we will need is functors. This is the category theoretic version of a homomorphism. A functor $F:\mathscr{C} \rightarrow \mathscr{D}$ consists of a mapping F_0 from the objects of $\mathscr{C}$ to those of $\mathscr{D}$, and another mapping F_1 from the arrows of $\mathscr{C}$ to those of $\mathscr{D}$, which preserves sources and targets, composition, and identities:

(51) a. If $f:A \rightarrow B$ is an arrow of $\mathscr{C}$, then $F_1(f)$ of $\mathscr{D}$ has source $F_0(A)$ and target $F_0(B)$.

b. For any arrows $f:A \rightarrow B$ and $g:B \rightarrow C$ of $\mathscr{C}$, $F_1(gf) = F_1(g)F_1(f)$.

c. For any object A of $\mathscr{C}$, $F_1(\mathbf{Id}_A) = \mathbf{Id}_{F_0(A)}$.

Once the introductory formalities are completed, people almost always drop the subscripts and just write $F(f):F(A) \rightarrow F(B)$, etc. The components of a functor are often described as 'mappings' since they may apply to bigger things than just sets, such as proper classes or worse, and so are not always functions, strictly speaking.

Like homomorphisms, functors and have inverses, and two categories connected by an inverse functors are said to be isomorphic. But it turns out that

the notion of isomorphism for categories is too strong to be really interesting; a more useful concept is 'equivalence' of categories, which we won't look at here.

A.4. Free CCCs and Indeterminates

The next step in our development is 'free' CCCs, possibly with 'indeterminates' adjoined, which are basically the same thing as the deductive categories we just introduced, perhaps with some additional rules of inference. But we'll construct them somewhat as if we hadn't seen the deductive ones before, to help consolidate the ideas.

First, objects (formulas). Given an arbitrary set of 'basic objects', we form objects recursively as follows:

(52) a. There is a terminal object $\mathbf{1}$ (different from any of the basic objects).

b. Given objects A, B, there is a product object $[A \times B]$ (the square brackets will usually be omitted, except when they distinguish between non-naturally-isomorphic objects).

c. Given objects A, B, there is an exponential object $[A \Rightarrow B]$. Products are taken to group more strongly then an exponentials, so that $[A \Rightarrow B \times C]$ notates $[A \Rightarrow [B \times C]]$.

Next, arrows. These will be developed in three stages, 'pre-pre-arrows' being the primitive ingredients (basic deductive steps), 'pre-arrows' being constructed by operations from pre-pre-arrows (deductions), and arrows formed as equivalence classes of pre-arrows (deductions equated by proof-normalization rules).

For the pre-pre-arrows of a CCC, we have:

(53) a. For each object A, an 'identity arrow' $\mathbf{Id}_A : A \to A$.

b. For each object A, a 'terminal arrow' $\bigcirc : A \to \mathbf{1}$.

c. For each pair of objects A, B, the left and right 'projection arrows' $\pi^1_{A,B} : A \times B \to A$, $\pi^2_{A,B} : A \times B \to B$.

d. For each pair of objects A, B, an 'evaluation arrow' $\mathbf{ev}^A_B : [[A \Rightarrow B] \times A] \to B$.

These can be construed as purely syntactic objects (expressions) with stipulated sources and targets.

We then construct the pre-arrows recursively as follows:

(54) a. Every pre-pre-arrow is a pre-arrow with the same source and target.

b. If $f{:}A \to B$, $g{:}B \to C$ are pre-arrows, then so their 'composition' $gf{:}A \to C$.

c. If $f{:}C \to A$, $g{:}C \to B$ are pre-arrows, then so is their 'pairing' $<f, g>{:}C \to A \times B$.

d. If $h{:}C \times A \to B$ is a pre-arrow, then so is its 'curry' $\mathbf{cur}(h){:}C \to [A \Rightarrow B]$.

To form the arrows of a CCC, we finally construct equivalence classes which are as fine as possible, subject to the requirement that the CCC equations are obeyed, which can be done by taking the intersection of all equivalence classes over the arrows that obey the following conditions (for all relevant objects A, B, C and arrows f, g, h):

(55) a. If $f{:}A \to B$, then $\mathbf{Id}_B f \equiv f \equiv f\mathbf{Id}_A$.

b. If $f{:}A \to B$, $g{:}B \to C$, $h{:}C \to D$, then $(hg)f \equiv h(gf)$ (for total notational finickiness, we should have included parentheses in the composition notation, but didn't).

c. For $f{:}C \to A$, $g{:}C \to B$, $\pi^1_{A,B}<f, g> \equiv f$ and $\pi^2_{A,B}<f, g> \equiv g$.

d. For $f{:}C \to A$, $g{:}C \to B$, $n{:}D \to C$, $<f, g>h \equiv <fh, gh>$.

e. For $h{:}C \times A \to B$, $\mathbf{ev}^A_B<\mathbf{cur}(h)\pi^1_{C,A}, \pi^2_{C,A}> \equiv h$.

f. For $k{:}C \to [A \Rightarrow B]$, $\mathbf{cur}(\mathbf{ev}^A_B<k\pi^1_{C,A}, \pi^2_{C,A}>) \equiv k$.

The arrows are the equivalence classes, with source and target those of their members.

Now we come to a new step, adjoining 'indeterminates', which, in terms of deduction, amounts to adding additional, ad-hoc inference rules. This is how lexical meanings can be handled. In the present construction, indeterminates

can be added by adding additional pre-pre-arrows at step (53). In the general case, an indeterminate can have any object as source and any as target, but we will only be interested in cases where the source is **1**, since these are an abstraction of elements of sets. This is the first of the two techniques discussed on LS:57, and the easier one, it seems to me. The second method amounts to adding indeterminates at the pre-arrow stage.[24]

A free CCC, with or without indeterminates, obeys only the equations specifically mandated by (55), so that for example if we have indeterminates $x{:}A \to B$, $y{:}B \to C$ and $z{:}A \to C$, it will not be the case that $z = yx$. This is the 'real reason' the 'functor lemma' below works.

A.5. The 'Functor Lemma'

The lemma can be stated as follows:

(56) Functor Lemma: If $\mathscr{C}$ is the free CCC over a collection of basic objects $\mathcal{B}$ and indeterminates $x_1, \ldots, x_n$, and $\mathscr{D}$ is a CCC, then any assignment F of objects of $\mathscr{D}$ to members of $\mathcal{B}$ and arrows of $\mathscr{D}$ to indeterminates of $\mathscr{C}$ that preserves sources and targets, determines a unique CC functor $\bar{\bar{F}}{:}\mathscr{C} \to \mathscr{D}$ that extends F.

This is really a generalization of LS' Proposition 5.1, p. 58, which only covers mappings of indeterminates, not basic objects.

We first need to extend the assignment of values to basic objects and indeterminates to composite objects, pre-arrows, and arrows. For objects and pre-arrows, this can be done recursively. Let F_0 be the assigment of objects of $\mathscr{D}$ to basic objects, F_1 the assignment of arrows of $\mathscr{D}$ to indeterminates. We can define $\bar{F}_0$ over all objects recursively as follows:

(57) a. If $B \in \mathcal{B}$, then $\bar{F}_0(B) = F_0(B)$.

b. $\bar{F}_0(A \times B) = \bar{F}_0(A) \times \bar{F}_0(B)$.

c. $\bar{F}_0([A \Rightarrow B]) = [\bar{F}_0(A) \Rightarrow \bar{F}_0(B)]$.

d. $\bar{F}_0(\mathbf{1}) = \mathbf{1}$ (the first in $\mathscr{C}$, the second in $\mathscr{D}$).

[24] Which seems unnecessarily confusing to me, since composition of the indeterminates with the pre-arrows then needs to be defined.

This is well-defined because every object of $\mathscr{C}$ is uniquely characterizable as a basic object, a product of two objects, or an exponential of two objects. (d) is congruent with (b-c) if we think of the terminal object as the output of a zero-place operation on objects.

For indeterminates, the requirement that F_1 be source- and target-preserving means that the following must be satisfied:

(58) For every indeterminate $x{:}A \to B$, $F_1(x){:}\bar{F}_0(A) \to \bar{F}_0(B)$.

(Although we're only really interested in indeterminates from $\mathbf{1}$, we'll cover the general case when it is easy to include). The other pre-pre-arrows are the CCC arrows, and for them, the requirements are essentially that the output of $\bar{F}_1$ be the corresponding CCC arrows on the appropriate objects:

(59) a. For every object A of $\mathscr{C}$, $F_1(\mathbf{Id}_A) = \mathbf{Id}_{\bar{F}_0(A)}$.

b. For every object A of $\mathscr{C}$, $F_1(\bigcirc_A) = \bigcirc_{\bar{F}_1(A)}$.

c. For all objects A, B of $\mathscr{C}$, $F_1(\pi^1_{A,B}) = \pi^1_{\bar{F}_0(A),\bar{F}_0(B)}$, and $F_1(\pi^2_{A,B}) = \pi^2_{\bar{F}_0(A),\bar{F}_0(B)}$.

d. For all objects A, B of $\mathscr{C}$, $F_1(\mathbf{ev}^A_B) = \mathbf{ev}^{\bar{F}_0(A)}_{\bar{F}_0(B)}$.

For the pre-arrows, we define $\bar{F}_1$ recursively, in the same manner as $\bar{F}_0$:

(60) a. If f is a pre-pre-arrow, then $\bar{F}_1(f) = F_1(f)$.

b. If $f{:}A \to B$, $g{:}B \to C$ are pre-arrows, then $\bar{F}_1(gf) = \bar{F}_1(g)\bar{F}_1(f)$.

c. If $f{:}C \to A$, $g{:}C \to B$ are pre-arrows, then $\bar{F}_1(\langle f, g\rangle) = \langle \bar{F}_1(f), \bar{F}_1(g)\rangle$.

d. If $h{:}C \times A \to B$ is a pre-arrow, then $\bar{F}_1(\mathbf{cur}(h)) = \mathbf{cur}(\bar{F}_1(h))$.

$\bar{F}_1$ clearly preserves sources and targets in the desired sense, and is well-defined because any given pre-arrow is uniquely analysable by the clauses of (60). But if $\mathscr{C}$ were to obey any equalities not required by the CC equations, then well-definedness would fail.

So finally we define $\bar{\bar{F}}_1$ on arrows by representatives. For this to work, every member of any given arrow must receive the same $\bar{F}_1$-value, which then becomes the $\bar{\bar{F}}_1$-value for the arrow. This requires showing that any

pre-arrows that are grouped into the same arrow by the equivalences of (55) have the same $\bar{F}_1$-value, which follows from the provisions of (59) plus the fact that the target category is a CCC, and so obeys these equations. I illustrate with (55d). I'll represent the equivalence class a.k.a. arrow of a pre-arrow by putting the pre-arrow in square brackets:

$$\begin{aligned}(61)\quad \bar{\bar{F}}_1([<f,g>h]) &= [\bar{F}_1(<f,g>h)]\\ &= [<\bar{F}_1(f),\bar{F}_1(g)>\bar{F}_1(h)]\\ &= [<\bar{F}_1(f)\bar{F}_1(h),\bar{F}_1(g),\bar{F}_1(h)>]\\ &= [\bar{F}_1(<fh,gh>)]\\ &= \bar{\bar{F}}_1([<fh,gh>])\end{aligned}$$

So we've finally got well-defined mapping $\bar{F}_0$ from the objects of $\mathscr{C}$ to those of $\mathscr{C}$, and $\bar{\bar{F}}_1$ from the arrows of $\mathscr{C}$ to those of $\mathscr{C}$; henceforth we'll forget about the bars and subscripts and just call the whole thing F.

So the next thing to do is to show that F is a CC functor. It is immediate that F preserves sources and targets, identities, and composition. To be a CC functor, F must also preserve the CC apparatus, which is again immediate from the way in which it has been set up. E.g.:

$$\begin{aligned}(62)\ \text{a.}\quad F([\pi^1_{A,B}]) &= [F(\pi^1_{A,B})]\\ &= [\pi^1_{F(A),F(B)}]\end{aligned}$$

$$\begin{aligned}\text{b.}\quad F([\mathbf{cur}(h)]) &= [F(\mathbf{cur}(h))]\\ &= [\mathbf{cur}(F(h))]\end{aligned}$$

Finally, for uniqueness, note that if G is a CC functor that agrees with F on basic objects and indeterminates, it will be recursively required to agree with F on everything.

This lemma allows us to define functors from our various free CCCs into **Sets** and each other, by specifying values for basic objects and indeterminates.

A.6. Functional Completeness and Internal Languages

I won't give a proof of the functional completeness theorem here, but will say a a bit about it does. Suppose we have a free CCC, $\mathscr{C}$, possibly with some indeterminates adjoined. We can adjoin another indeterminate x of

type A to get the category $\mathscr{C}[x]$. Now let $\phi{:}\mathbf{1} \to B$ be an arrow in $\mathscr{C}[x]$, which might or might not contain x (i.e. might only be in $\mathscr{C}[x]$, or also in $\mathscr{C}$). This is commonly notated by writing ϕ as $\phi(x)$, but I won't do that here.

The theorem says that there is a unique arrow $f{:}A \to B$ in (the original) $\mathscr{C}$ such that $\phi = fx$. If we think of x as a contaminating impurity in ϕ, the idea is that we can squeeze it into a single position, leaving the healthy residue f. Suppose for example that $\phi = y$, where y is some indeterminate in $\mathscr{C}$ of type B. Then $f = y\bigcirc_A$. Since the arrow f that is in $\mathscr{C}$ but not $\mathscr{C}[x]$ is unique, we can designate it with a term, such as $\lambda_{x\in A}\phi$. Conceptually, this is more like a combinatory logic term that a conventional lambda-term interpreted by means of assignments. However, it works out to be more convenient overall to let $\lambda_{x\in A}\phi$ be an abbrevation for the 'name of' f, that is $\ulcorner f \urcorner{:}\mathbf{1} \to [A \Rightarrow B]$.

This allows for a smooth formulation of the expressions of the 'internal language' of the CCC, which is a system of expressions designating arrows from $\mathbf{1}$ built out of the ingredients of the category itself, so that its denotation function is the identity (if we treat the lambda-expressions as convenient abbreviations). So if for example $\lambda_{x\in A}\phi$ and $a{:}\mathbf{1} \to A$ are expressions/arrows in the internal language, then so is $\mathbf{ev}^A_B{<}\lambda_{x\in A}\phi, a{>}$, denoting the effect of applying the function/arrow designated by $\lambda_{x\in A}\phi$ to a.

Appendix B Table of Symbols

$\mathscr{G}$	Glue CCC	section 5.
$\mathscr{L}$	Lexical CCC	section 2.
$\mathscr{P}$	Proof CCC	section 3.
$\mathscr{S}$	Semantics CCC	section 2.
τ	substitution from basic types of $\mathscr{P}$ to those of $\mathscr{G}$	section 3.
$I_{\mathscr{L}}$	Interpretation functor from $\mathscr{L}$ to $\mathscr{S}$	section 2.
LS	Lambek and Scott (1986)	
$\mathrm{NJ}_{\to,\wedge}$	Intuitionistic $\to, \wedge$ logic	section 3.

References

Alsina, A. 1996. *The Role of Argument Structure in Grammar*. Stanford, CA: CSLI Publications.

Andrews, A. D. 1976. The VP complement analysis in Modern Icelandic. In *Montreal Working Papers in Linguistics*. reprinted in Maling and Zaenen (1990).

Andrews, A. D. 1982. The representation of case in Modern Icelandic. In Bresnan (Ed.).

Andrews, A. D. 1990. Unification and morphological blocking. *Natural Language and Linguistic Theory* 8:507–557.

Andrews, A. D. 2007. Glue semantics for clause-union complex predicates. In M. Butt and T. H. King (Eds.), *The Proceedings of the LFG '07 Conference*, 44–65. Stanford CA: CSLI Publications. `http://cslipublications.stanford.edu/LFG/12/lfg07.html` (accessed Feb 19, 2010).

Andrews, A. D. 2008. Propositional glue and the correspondence architecture of LFG. to appear in Linguistics and Philosophy. Draft uploaded to semantics archive; also `http://arts.anu.edu.au/linguistics/People/AveryAndrews/Papers`.

Andrews, A. D. 2010. 'Grammatical' vs. 'lexical' meaning constructors for glue semantics. In Y. Treis and R. D. Busser (Eds.), *Selected Papers from the 2009 Conference of the Australian Linguistic Society*. The Australian Linguistic Society. URL: `http://www.als.asn.au/proceedings/als2009.html`, also `http://arts.anu.edu.au/linguistics/People/AveryAndrews/Papers`.

Aoto, T., and H. Ono. 1994. Uniqueness of normal proofs in the $\{\rightarrow, \wedge\}$ fragment of NJ. Research Report IS-RR-04-0024F, School of Information Science, JAIST (reference gotten from `http://www.nue.riec.tohoku.ac.jp/user/aoto/`).

Asudeh, A. 2004. *Resumption as Resource Management*. PhD thesis, Stanford University, Stanford CA. http://http-server.carleton.ca/~asudeh/ (accessed 19 Feb 2010).

Asudeh, A. 2006. Direct compositionality and the architecture of LFG. In M. Butt, M. Dalrymple, and T. H. King (Eds.), *Intelligent Linguistic Architectures: Variations on Themes by Ronald M. Kaplan*, 363–387. Stanford, CA.

Asudeh, A. 2008. Towards a unified theory of resumption. URL: http://http-server.carleton.ca/, http://semanticsarchive.net/Archive/jNjZWRkM/asudeh08-rp.pdf.

Asudeh, A., and I. Toivonen. 2009a. Copy-raising and perception. To appear in Natural Language and Linguistic Theory. Submitted draft; URL: http://http-server.carleton.ca/~asudeh/.

Asudeh, A., and I. Toivonen. 2009b. Lexical-functional grammar. In B. Heini and N. Narrog (Eds.), *The Oxford Handbook of Linguistic Analysis*. Oxford University Press. URL: http://http-server.carleton.ca/~asudeh/research/index.html.

Babaev, A., and S. Solov'ev. 1982. A coherence theorem for canonical morphisms in cartesian closed categories. *Zap. Nauchn. Sem* LOM1 88:3–39, 236. also J. Soviet Math. 20:2263-2279.

Barr, M., and C. Wells. 1999a. *Category Theory for Computing Science*. Montréal: Centre de Recherches Mathématiques. 3rd edition. can be ordered from: http://crm.umontreal.ca/pub/Ventes/CatalogueEng.html. The most essential content for linguists, but no exercises, can be found in Barr and Wells (1999b), available free online.

Barr, M., and C. Wells. 1999b. Category theory lecture notes for ESSLLI. URL: http://www.let.uu.nl/esslli/Courses/barr-wells.html. a condensed version, without exercises, of Barr and Wells (1999a). It is impressively well targeted on material that seems likely to be useful to linguists.

Boeckx, C., N. Hornstein, and J. Nunes. 2010. Icelandic control really is A-movement: Reply to Bobaljik and Landau. *Linguistic Inquiry* 41:111–130.

Bresnan, J. W. (Ed.). 1982. *The Mental Representation of Grammatical Relations*. Cambridge MA: MIT Press.

Crouch, R., and J. van Genabith. 1999. Context change, underspecification, and the structure of glue language derivations. In Mary Dalrymple (Ed.), *Syntax and Semantics in Lexical Functional Grammar: The Resource-Logic Approach*, 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., R. M. Kaplan, J. T. Maxwell, and A. Zaenen (Eds.). 1995. *Formal Issues in Lexical-Functional Grammar*. Stanford, CA: CSLI Publications. URL: `http://standish.stanford.edu/bin/detail?fileID=457314864`.

de Groote, P. 1999. An algebraic correctness criterion for intuitionistic multiplicative proof-nets. *Theoretical Computer Science* 224:115–134. `http://www.loria.fr/~degroote/bibliography.html` (accessed 19 Feb 2010).

Girard, J.-Y., Y. Lafont, and P. Taylor. 1989. *Proofs and Types*. Cambridge: Cambridge University Press. `http://www.monad.me.uk/stable/prot.pdf` (accessed 19 Feb 2010).

Jackendoff, R. S. 1972. *Semantic Interpretation in Generative Grammar*. MIT Press. extensive revision of 1969 MIT dissertation.

Jackendoff, R. S. 1990. *Semantic Structures*. MIT Press.

Johnson, M. 1999. Type-driven semantic intepretation and feature dependencies in R-LFG. In M. Dalrymple (Ed.), *Syntax and Semantics in Lexical Functional Grammar: The Resource-Logic Approach*, 359–388.

Kaplan, R. M. 1995. The formal architecture of LFG. In M. Dalrymple, R. M. Kaplan, J. T. Maxwell, and A. Zaenen (Eds.), *Formal Issues in Lexical-Functional Grammar*, 7–27. CSLI Publications.

King, J. C. 2007. *The Nature and Structure of Content*. Oxford University Press.

Klein, E., and I. Sag. 1985. Type-driven translation. *Linguistics and Philosophy* 8:163–201.

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*. Stanford, CA: CSLI Publications.

Kokkonidis, M. 2007d. Towards a Type-Logical Lexical Functional Grammar. In M. Butt and T. King (Eds.), *Proceedings of LFG 2007*. Stanford, CA: CSLI Publications.

Kokkonidis, M. 2008. First order glue. *Journal of Logic, Language and Information* 17:43–68. First distributed 2006; URL: `http://citeseer.ist.psu.edu/kokkonidis06firstorder.html`.

Lambek, J., and P. J. Scott. 1986. *Introduction to Higher Order Categorical Logic*. Cambridge University Press.

Lev, I. 2007. *Packed Computation of Exact Meaning Representations using Glue Semantics (with automatic handling of structural ambiguities and advanced natural language constructions)*. PhD thesis, Stanford University. URL: `http://sites.google.com/site/pulcproject/material` (viewed July 2010).

Maling, J., and A. Zaenen. 1990. *Modern Icelandic Syntax*. Academic Press.

Manning, C. D. 1996. *Ergativity: Argument Structure and Grammatical Relations*. Stanford CA: CSLI Publications. originally Stanford PhD dissertation, 1994.

Marantz, A. 1984. *On the Nature of Grammatical Relations*. Cambridge MA: MIT Press.

Moot, R. 2002. *Proof-Nets for Linguistic Analysis*. PhD thesis, University of Utecht. URL: `http://www.labri.fr/perso/moot/` (viewed June 2008), also `http://igitur-archive.library.uu.nl/dissertations/1980438/full.pdf`.

Muskens, R. 2007. Intensional models for the theory of types. *Journal of Symbolic Logic* 72:98–118.

Nordlinger, R. 1998. *Constructive Case*. Stanford CA: CSLI Publications.

Oehrle, D. 1999. LFG as labeled deduction. In M. Dalrymple (Ed.), *Semantics and Syntax in Lexical Functional Grammar: The Resource Logic Approach*. Cambridge, MA: The MIT Press.

Partee, B. H. 2006. Do we need two basic types. URL: `http://people.umass.edu/partee/docs/`.

Perrier, G. 1999. Labelled proof-nets for the syntax and semantics of natural languages. *L.G. of the IGPL* 7:629–655. `http://www.loria.fr/~perrier/papers.html` (accessed 19 Feb 2010).

Pierce, B. 1991. *Basic Category Theory for Computer Scientists*. MIT Press.

Pollard, C. 2008. Hyperintensions. *Journal of Logic and Computation* 18:257–282. `http://www.ling.ohio-state.edu/~pollard/cvg/hyper.pdf` (accessed 19 Feb 2010).

Restall, G. 2000. *An Introduction to Substructural Logics*. Routledge.

Simpson, J. H. 1991. *Warlpiri Morpho-Syntax*. Kluwer Academic.

Troelstra, A., and H. Schwichtenberg. 2000. *Basic Proof Theory*. Cambridge University Press. 2nd edition.

Zaenen, A., J. Maling, and H. Thráinsson. 1985. Case and grammatical functions: the Icelandic Passive. *Natural Language and Linguistic Theory* 3:441–83.