The Calculus of Responsibility and Commitment

Carl Pollard
INRIA-Lorraine and Ohio State University

1 Introduction

Ever since Montague (1974 [1970]) laid the foundations for formally precise analysis of natural language (hereafter NL) semantics in the late 1960's, the typed lambda calculus (hereafter TLC) and certain of its extensions[1] have been the linguists' tool of choice for representing the meanings of NL expressions. But starting around the turn of the millenium, motivated by a range of linguistic phenomena collectively known as **covert movement** phenomena[2], logical grammarians of various persuasions have proposed the use of other semantic term calculi that embody, directly or indirectly, some notion or other of **continuation**.

In this paper, I will do the same thing. My justification for stepping into the fray, in spite of my distinctly amateur standing vis-à-vis the mathematics and computer science of continuations, is that the proposals I have seen so far seem, for various reasons, not yet able to compete with conventional Montague-style compositional semantics in the linguistic marketplace. Some of the proposals appear to be incompletely specified; others require technical knowledge of mathematics or computer science that almost no linguists control (or even have straightforward access to); and in others, the sheer complexity seems incommensurate with the difficulty of the problems to be solved. At the heart of my proposal is a term calculus called RC which is suggested by way of a replacement for TLC as a notational system for NL meanings. It is my intention that RC be as easy for linguists to learn and use as TLC is, while at the same time doing the semantic heavy lifting performed by existing systems of continuized semantics.

This paper is highly programmatic in nature. For more of the linguistic motivation and analysis of linguistic examples, the reader is referred to the

[1] Specifically, variants of Church's (1940) and Henkin's (1950) simple theory of types, especially Montague's own Intensional Logic (IL) and Gallin's (1975) Ty2.

[2] So-called because they are analyzed by transformational generative grammarians via tree operations on syntactic representations which are reflected in the **logical form (LF)** of the sentence being analyzed, but which take place too late in the derivation to be reflected in the **phonetic form (PF)**. By no means does the use of the term 'covert movement' imply advocacy of the technology of transformational grammar, such as arboreal structural representations of linguistic expressions (not to mention derivations upon such representations made up of structural operations that delete, copy, or move subtrees.

companion piece (Pollard in press).

2 Background

The so-called covert movement phenomena involve a linguistic expression that seems, pretheoretically speaking, to have semantic scope, but to occupy a syntactic position appropriate not for an operator with such a scope, but rather, for a variable bound by such an operator. In the parlance of transformational grammar, the operator remains, in overt syntax, **in situ**, and moves to its scope position only covertly, leaving in its place a variable which it binds at the level of LF.

Examples of in situ operators include (but are by no means limited to) the following:

(1) **Quantificational Noun Phrases**
Kim thinks [**everyone** walks].

Quantifer scope ambiguity: the QNP **everyone** can take scope in either the root clause or the embedded clause.

(2) **In-situ interrogative expressions (wh-in situ)**
Who asked [who ordered **what**]?

Ambiguity of wh-in-situ construal: the in-situ wh-expression **what** can scope in either the root question or the embedded question ('Baker's ambiguity').

(3) **Pied Piping**

a. [To **whose** parents]$_i$ were you speaking t_i?

b. These are the reports [the height of the lettering on the covers of **which**]$_i$ the government prescribes t_i.

The 'pied piped' (interrogative or relative) expression is *in situ* relative to a larger expression which undergoes 'overt movement'.

(4) **Comparative Word (more/less) or Morpheme (-er)**
Russell thought [the yacht was long**er**] than it was.

Ambiguity of comparative operator construal: the comparative morpheme **-er** can scope in either the root question (how long Russell thought the yacht was exceeded the actual length) or in the embedded clause (Russell stupidly thought the yacht's length was self-exceeding).

(5) **Focus**

Mary thinks that **John**[H*] will vote for McCain.

The focal or 'A' [H*] pitch accent on **John** somehow conveys that there is at least one contextually salient individual different from John who Mary does *not* think will vote for McCain.

(6) **In-situ Contrastive Topic**

He thinks she likes **bagels**[L+H*].

The contrastive-topic or 'B' [L+H*] pitch accent on **bagels** somehow conveys that whether he thinks she likes certain things, including bagels, is at issue, and that the answers may vary.

(7) **Phrasal-Comparative Associates**

a. **The Pope** goes to Minneapolis more often than St. Paul. (St. Paul = the saint)

b. The Pope goes to **Minneapolis** more often than St. Paul. (St. Paul = the city)

Roughly speaking, the pitch-accented phrase in the main clause (the **associate**) and the complement phrase to *than* (the **remnant**) are plugged into the same context and then the least upper bounds of the two resulting sets of degrees compared.

(8) **Superlative Associates**

a. **Kim** likes Sandy the most. (Kim likes Sandy more than anyone else likes Sandy.)

b. Kim likes **Sandy** the most. (Kim likes Sandy more than Kim likes anyone else.)

c. **Kim** likes **Sandy** the most. (Kim likes Sandy more than anyone else likes anyone else.)

A superlative sentence asserts that the interpretation of the (possibly discontinuous) associate is where a certain degree-valued function assumes its maximum value.

If we call the linguistic material surrounding the in situ operator its **linguistic context** and that part of the linguistic context which expresses the operator's scope the **(delimited) continuation**[3] of the operator, then

[3]Here, the sense of 'delimited' is that the scope of the operator need not be the entire linguistic context (the root clause).

it is easy to see why Barker (2002), who pioneered continuation semantics for NL, asserted what he called the Continuation Hypothesis:

(9) **Barker 2002**

a. **The Continuation Hypothesis:** Some linguistic expressions (in particular, QNPs) have denotations that manipulate their own continuations.

b. Barker shows how to transform a CFG with conventional Montague-style compositional semantics into one with 'continuized semantics', by analogy with Fischer's (1972)/Plotkin's (1975) **call-by-value CPS transform**.

c. In terms, variables with overbars are called 'continuation variables' and certain implicative types with result type t are called 'continuation types'.

d. *Formally*, these are ordinary variables and ordinary types; the semantic calculus is embedded into ordinary (intuitionistic) TLC.

(10) **de Groote 2001**

a. de Groote uses a nonconfluent calculus with an involutive negation (similar in some respects to Parigot's (2000) **symmetric $\lambda\mu$-calculus**), with Montague's type t as $\bot$, to analyze one example of quantifier scope ambiguity.

b. Unlike Barker's CPS approach, this one (like those of Shan and Bernardi/Moortgat mentioned below, is in 'direct-style'.

c. Though it is claimed that $\lambda\mu$-calculus "allows Cooper's (1983) storage to be given a type-logical foundation", it is not clear how to generalize this approach to operators whose scope has type different from t, or whose result type differs from the scope type.

(11) **Shan 2004**

a. Shan pioneered the use of **delimited continuations** (Felleisen 1988) in NL.

b. The term calculus ('logical metalanguage') mixes syntactic and semantic constructs (for example it has directional applications and abstractions).

c. The ternary type constructor A^C_B provides the types for operators that 'scope over a B to bind an A-variable, resulting in a C.'

d. This constructor is a semantic analog of Moortgat's (1996) q-constructor.

e. A^C_B is mapped by the CPS transform to $(A \to B) \to C$.

f. Besides familiar TLC and Lambek-calculus constructs, the language employs a **hierarchy of control operators** (generalized shift and reset supercripted with 'strength levels', Danvy and Filinsky 1990).

(12) **Bernardi and Moortgat 2007**

a. Curien and Herbelin's (2000) $\bar{\lambda}\mu\tilde{\mu}$**-calculus** "is to sequent calculus as [the original, not the symmetric] $\lambda\mu$-calculus [Parigot 1992] is to natural deduction."

b. Bernardi and Moortgat develop a linear, directional variant of $\bar{\lambda}\mu\tilde{\mu}$-calculus to provide a semantic interpretation for **symmetric categorial grammars** based on the Lambek-Grishin calculus (Moortgat 2007).

c. Unlike other systems of continuation semantics (and predecessors such as Cooper 1983 and Hendriks 1993), but like mainstream categorial grammar, the Bernardi-Moortgat semantics is **functional** in the sense that interpretation is a function from syntactic proofs to meanings.

3 Cooper Storage and Retrieval

Unlike the approaches mentioned in the previous section, the inspiration for the RC calculus comes not from computer science but from linguistics, more specifically the storage-and-retrieval approach to scope developed by Robin Cooper (1975, 1983). The basic idea is easy to describe in a way that a first-year linguistic graduate student with a basic understanding of TLC can grasp almost immediately.

(13) **Cooper Storage and Retrieval (Intuitively)**

a. Suppose that while semantically interpreting a syntactic derivation bottom up, you encounter an in-situ operator with semantics a of semantic type $(A \to B) \to C$. Then (**storage**) you can replace a by a variable $x : A$ and place the pair (a, x) in the store.

b. Stores 'percolate upward' through derivations.

c. If you get to a node in the derivation with semantics $b : B$, then (**retrieval**) you can remove (a, x) from the store and replace $b : B$ by $a(\lambda_x b) : C$.

d. In case the in-situ operator is a QNP, $A = \mathrm{e}$, and $B = C = \mathrm{t}$.

(14) **Sociology of Cooper Storage and Retrieval**

a. Cooper didn't describe it this way. He expressed it all in terms of Montagovian model theory.

b. Cooper was at pains to avoid using λ-calculus, to make it clear he was not advocating some form of LF.

c. Even though this technology makes syntax much simpler (e.g. QNPs are just NPs), the resulting nonfunctionality of semantic interpretation was widely viewed with alarm.

d. However it was embraced in some quarters (e.g. Bach and Partee 1980, and in HPSG).

e. Categorial grammarians tend to describe this technology as 'noncompositional', 'baroque', 'ad hoc', 'a mess', 'as bad as covert movement', etc. (some of the more charitable characterizations).

f. The schemes of continuized semantics mentioned above are presented as major improvements on Cooper storage and retrieval.

(15) **Why RC Calculus?**

a. In spite of its reputation, Cooper's basic approach to scoping in situ operators is simple, illuminating, and easy to grasp.

b. But Cooper's insistence on a model-theoretic presentation made storage look more complicated than it really was.

c. RC calculus is designed to render the intuitions underlying Cooper storage more immediately graspable, via a purely syntactic presentation.

(16) **Toward RC Calculus**

a. RC is a term calculus in the labelled Gentzen-sequent style of natural deduction.

b. As far as I have been able to discern so far, RC can do all the linguistic work that has been done so far with continuized semantics.

c. RC can be directly semantically interpreted, but the easiest way is to transform RC meaning terms into TLC.

d. This transform is the RC analog of CPS transforms, but much simpler.

e. Morover RC meaning terms look familiar to people used to LF.

4 RC Types, Terms, and Commitments

Like TLC, RC has types, terms, and typing judgments. Also like TLC, the variable environment of an RC typing judgment is just a set of variable/type pairs to the left of the turnstile, which we will call the **variable context**; but an RC variable environment additionally has a **Cooper store**, written to the right and demarcated by a **co-turnstile** $\dashv$:

(17) **Format for RC Typing Judgments**
$\Gamma \vdash a : A \dashv \Delta$

The Cooper store is also called the **variable co-context**; the 'co-' here is mnemonic not only for 'Cooper'; but also for 'Commitment' (for reasons to be explained soon), for 'Covert Movement', and for 'Continuation' (since the operators stored in them will scope over their own continuations). Thus a judgment like (17) is read 'the term a is assigned the type A in the context Γ and the co-context Δ.'

(18) **RC Types**

a. There are some **basic** types.

b. If A and B are types, then $A \rightarrow B$ is an **implicative** type with **antecedent** type A and **consequent** type B.

c. If A, B, and C are types, then $\mathrm{O}[A, B, C]$, usually abbreviated (following Shan 2004) to A^C_B, is an **operator** type with **binding** type A, **continuation** type (or **scope** type) B, and **result** type C.

(19) **Basic Types**
Here, for expository simplicity, just Montague's extensional types e and t. But to do serious semantics, we really need to work in an intensional (or hyperintensional) semantic type theory.

(20) **Implicative Types**
No surprises here.

(21) **Operator Types**

a. These will be the semantic types for expressions which would be analyzed in TG as undergoing $\bar{\text{A}}$-movement (either overt or covert).

b. The O-constructor is essentially like Moortgat's (1996) q-constructor, with the crucial difference that it belongs to the *semantic* logic, *not* the syntactic one.

c. Thus, for example, while for Moortgat (1996) a QNP would have syntactic category q[NP, S, S] and semantic type (e → t) → t, in RC it has syntactic type (simply) NP and semantic type $\mathrm{e}_{\mathrm{t}}^{\mathrm{t}4}$

d. Operator types are similar to Shan's (2004) **impure** types.

(22) **RC Terms**

a. There is a denumerable infinity of **variables** of each type.

b. There can be finitely many **basic constants** of each type.

c. There are **functional** terms of the form $(f\ a)$, where f and a are terms.

d. There are **binding** terms of the form $(a_x b)$ where a and b are terms and x is a variable.

e. But there is no λ binder.

(23) **What goes into the Cooper Store?**

a. The Cooper stores (co-contexts) will contain operators to be scoped, each paired with the variable that it will eventually bind.

b. We call such stored pairs **commitments**, and write them in the form a_x, where the type of x is the binding type of a.

c. Then we call x a **committed** variable, and say that a is **committed** to bind x.

d. By contrast, the variables in the (left-of-turnstile) context are called **uncommited** variables.

We can now give the rules of RC.

5 RC Rules

(24) **Schema H (Hypotheses)**

$x : A \vdash x : A \dashv$ (x fresh)

a. This is the usual ND schema for positing hypotheses.

[4] Actually QNPs have to be polymorphically typed as $\mathsf{e}_{\mathsf{t}_\sigma}^{\mathsf{t}_\sigma}$ where σ ranges over strings of types and $A_\epsilon =_{\mathrm{def}} A$, $A_{B\sigma} =_{\mathrm{def}} B \to A_\sigma$. This is necessary to account for the fact that QNPs can scope not only over propositions, but also over propositional functions (e.g. ones of type e → t, as in *Kim tried [to find a unicorn)* or *Mia knows every [owner of a hash bar]*). I am grateful to Patrick Blackburn and Scott Martin for alerting me to this issue. It does not arise in HPSG because VPs and $\bar{\mathrm{N}}$s have propositional semantics (since arguments are incorporated into predicates by structure-sharing, not by Modus Ponens).

b. It is the sole mechanism provided for introducing uncommitted variables into a semantic derivation.

c. We presuppose a syntax-semantics interface that recursively specifies a set of syntactic-semantic **derivation pairs** (Pollard in press). An uncommitted semantic variable will always be paired with a trace (syntactic variable) [5].

(25) **Schema A (Nonlogical Axioms)**

$\vdash c : A \dashv$ (c a basic constant of type A)

a. The basic constants notate lexical meanings, not just of words but also of bound morphemes that function as basic syntactic elements, such as argument clitics, phrasal affixes, sentence particles, etc. (some of which might have intonational or even empty phonological realizations).

b. The rules of the syntax-semantics interface that pair basic syntactic entities with their meanings are the **lexicon**.

(26) **Schema M (Modus Ponens)**

If $\Gamma \vdash f : A \rightarrow B \dashv \Delta$ and $\Gamma' \vdash a : A \dashv \Delta'$
then $\Gamma;\Gamma' \vdash (f\ a) : B \dashv \Delta;\Delta'$

a. This is the usual ND Modus Ponens, except that co-contexts as well as contexts have to propagated from premisses to conclusions (cf. (13b)).

b. Semicolons in (co-)contexts represent set union (necessarily disjoint, since variables are always posited fresh).

c. The syntax-semantics interface will pair instances of M with syntactic **merges**.

(27) **Schema C (Commitment)**

If $\Gamma \vdash a : A^C_B \dashv \Delta$ then $\Gamma \vdash x : A \dashv a_x : A^C_B;\Delta$ (x fresh)

a. This is a straightforward ND formulation of Cooper storage (13a).

b. It generalizes Carpenter's (1997) ND Introduction rule for Moortgat's (1988) $\Uparrow$ (essentially the special case of q where the scope type

[5] This is in keeping with an overall **parallel** grammar architecture, as opposed to the **cascaded** achitecture of mainstream generative grammar, which requires that, somehow or other, traces be replaced by 'logical variables' during the derivation from overt syntax to LF.

and the result type are the same), but in the semantics, **not** in the syntax.

(28) **More about Schema C**

a. The type of a committed variable always matches the binding type of the operator it is committed to.

b. The syntax-semantics interface will guarantee that when an operator gets committed in the semantic derivation, **no corresponding syntactic change takes place**.

c. This is one of two reasons why the relation between syntax and semantics is **not** a function.

d. In this respect, our proposed architecture for the syntax-semantics interface resembles Cooper 1975/1983, Hendriks 1993, and HPSG, as well as most of the continuized-semantics proposals.

e. This sets it apart from mainstream categorial grammar, as well as the Bernardi-Moortgat functional style of continuized semantics.

(29) **Schema R (Responsibility)**

If $\Gamma \vdash b : B \dashv a_x : A^C_B; \Delta$ then $\Gamma \vdash (a_x b) : C \dashv \Delta$
(x free in b but not in Δ)[6]

a. This is a straightforward ND formulation of Cooper retrieval (13c).

b. It generalizes Carpenter's (1997) Elimination rule for Moortgat's $\Uparrow$, but, again, in the semantics, **not** in the syntax.

c. It is called Responsibility because it is about fulfilling commitments.

d. As with Commitment, the syntax-semantics interface will ensure that instances of Responsibility correspond to no syntactic change.

e. This is the other reason why the relation between syntax and semantics is **not** a function.

(30) **Schema O (Outside Assistance)**

If $\Gamma \vdash a : A^C_B \dashv \Delta$ and $x : A, \Gamma' \vdash b : B \dashv \Delta'$
then $\Gamma; \Gamma' \vdash (a_x b) : C \dashv \Delta, \Delta'$

The schema is so-called because if you cannot discharge your responsibilities (to make sure all variables eventually get bound) on your own

[6]This side condition and the one for Schema C express the **linearity** of co-contexts. They rule out illicit retrievals of the kind that motivated Keller's (1988) nested stores (see Blackburn and Bos pp. 122-125 for discussion).

(as the in-situ operators within you fulfil their commitments), then you have to get help from somewhere else.

(31) **More about Schema O**

a. O is also mnemonic for Overt Movement, because in TG the expression whose meaning is a would be analyzed as having moved (in overt syntax) from the position occupied by the trace (which somehow gets converted to x in between overt syntax and LF).

b. Note the strong similarity of the conclusions in Schemas R and O: in both cases the operator a binds the x in b to produce a C.

c. The difference is that in the former (in situ binding) case, b is a's continuation.

d. The idea for this schema was triggered by Guy Perrier's recent observation (p.c.) that the Topicalization Rule in Pollard 2007 was an elimination rule for the movement flavor of implication (SLASH, similar to the $\uparrow$ connective of Bach 1981 and Moortgat 1988).

(32) **Schema O vs. Anoun and Lecomte's $\multimap$ IE**

a. The $\multimap$ IE schema and Schema O have the same philosophy: to 'encapsulate' implication introductions within rules that immediately eliminate the implication.

b. But $\multimap$ IE has $(A \multimap B) \multimap C)$ in place of A^C_B, and $a(\lambda_x b)$ in place of $(a_x b)$. A problem with that formulation is that not all expressions with types of the form $(A \multimap B) \multimap C$ can undergo 'overt movement'.

c. In $\multimap$ IE, *controlled* hypotheses constrain extraction paths analogously to the use of functional uncertainty in LFG.

d. $\multimap$ IE is inspired by Vermaat's (1999) multimodal CG reformulation of Stabler's (1999) computational embodiment of Chomsky's (1995) MP.

e. Whereas Schema O is inspired by Gazdar's (1979) linking schemata for topicalization, *wh*-relatives, and *wh*-questions (his (28), (51), and (57), via GPSG and HPSG.

f. Unfortunately the feature-structural encoding of HPSG obscured the fact that it was essentially (albeit unknowingly) a natural-deduction system.

6 Conclusion

There are still a few loose ends to tie up. For one thing, so far we have not analyzed even one example. Since this paper is already getting overlong for a workshop paper, we offer just the obligatory QNP scope-ambiguity example and refer to Pollard in press for other examples (and for an explication of the Convergent Grammar (CVG) framework that the semantic approach sketched here is intended to work with).

(33) **Obligatory QNP Scope Ambiguity Example**

a. Kim thinks everyone walks.

b. CVG syntactic analysis:
$\vdash$ ($^{\text{S}}$ Kim (thinks ($^{\text{S}}$ everyone walks) $^{\text{C}}$)) : S

All the subterms here are instances of the CVG syntactic schemas SM (Subject Merge) or CM (Complement Merge). The absence of a co-turnstile here is not a typo: syntactic derivations do not have co-contexts!

c. Narrow-scope semantic analysis:
$\vdash$ ((think' (everyone'$_x$(walk' x))) Kim') : t $\dashv$

d. Wide-scope semantic analysis:
$\vdash$ everyone'$_x$ ((think' (walk' x)) Kim') : t $\dashv$

Second, what do these RC terms really mean? The easiest way to explain this is to translate them into TLC, since everybody knows how to semantically interpet that. Fortunately, the translation from RC to TLC is a considerably simpler than the CPS transforms of (say) the $\bar{\lambda}\mu\tilde{\mu}$-calculus:

(34) **From RC to TLC**

a. First make sure the term is responsible (the Cooper store is empty), because commitments have no translation into TLC.

b. Replace every operator type A^C_B by $(A \to B) \to C)$, and every binding subterm $(a_x b)$ by $(a\ \lambda_x b)$.[7]

c. By convention we write application terms as $(f\ a)$ in RC and as $f(a)$ in TLC, just to make it easy to tell at a glance which calculus the term belongs to.

[7]For polymorphically typed operators, this is only for the lowest (ϵ) types. For $\sigma = A_0 \ldots A_n$ $(n > 0)$, the transform is $\lambda_{x_0} \ldots \lambda_{x_n}(a\ \lambda_x b(x_0) \ldots (x_n))$.

For example, the TLC translations of the RC terms in (33) are as expected:

(35) **TLC Terms for Scope Ambiguity Example**

a. Narrow-scope semantic analysis:
$\mathsf{think'}(\mathsf{everyone'}(\lambda_x \mathsf{walk'}(x)))(\mathsf{Kim'}) : \mathrm{t}$

b. Wide-scope semantic analysis:
$\mathsf{everyone'}(\lambda_x \mathsf{think'}(\mathsf{walk'}(x))(\mathsf{Kim'})) : \mathrm{t}$

Finally, since (in case it is still not obvious) the name RC is a (minimally) veiled reference to Robin Cooper, I want to emphasize that when he first developed his theory of storage and retrieval, Cooper by no means advocated a term-calculus formulation of the theory, quite the contrary in fact. So it's anybody's guess whether he will mind my naming a proof-theoretic embodiment of his theory after him. But since he has been working in Martin-Löf type theory for a number of years now, I like to think tht perhaps he won't mind too much.

Ackowledgments

For helpful discussion and comments on earlier stages of this work, I am grateful to Chris Barker, Patrick Blackburn, Wojciech Buszkowski, Robin Cooper, David Dowty, Philippe de Groote, Ruth Kempson, Brad Kolb, Oleg Kiselyov, Yusuke Kubota, Alain Lecomte, Tim Leffel, Jim Lambek, Scott Martin, Michael Moortgat, Glyn Morrill, Reinhard Muskens, Guy Perrier, Andy Plummer, Ken Shan, Elizabeth Smith, Chris Worth, workshop participants at the ESSLLI 2007 Workshop on New Directions in Type-Theoretic Grammar (Dublin), the Fourth Workshop on Lambda Calculus and Formal Grammar (Nancy, 2007), the Colloque en l'honneur d'Alain Lecomte (Pauillac, 2007), the Second Workshop on Types, Logic, and Grammar (Barcelona, 2007), and colloquium audiences at the Séminaire de l'UMR 7023, CNRS/Université de Paris 8 (2008) and the Centre de Lingüística Teòrica, Universitat Autònoma de Barcelona (2008). For their help in providing the conditions that made this research possible, I am grateful to Philippe de Groote, Carlos Martin Vide, and to the Department of Linguistics and College of Humanities of Ohio State University. Some of the research reported here was supported by grant no. 2006PIV10036 from the Agència de Gestió d'Ajuts Universitaris i de Recerca of the Generalitat de Catalunya.

Appendix: The RC Schemata

H: $x : A \vdash x : A \dashv$ (x a fresh variable of type A)

A: $\vdash c : A \dashv$ (c a basic constant of type A)

M: If $\Gamma \vdash f : A \to B \dashv \Delta$ and $\Gamma' \vdash a : A \dashv \Delta'$
then $\Gamma; \Gamma' \vdash (f\ a) : B \dashv \Delta; \Delta'$

C: If $\Gamma \vdash a : A^C_B \dashv \Delta$ then $\Gamma \vdash x : A \dashv a_x : A^C_B; \Delta$ (x fresh)

R: If $\Gamma \vdash b : B \dashv a_x : A^C_B; \Delta$ then $\Gamma \vdash (a_x b) : C \dashv \Delta$

O: If $\Gamma \vdash a : A^C_B \dashv \Delta$ and $x : A, \Gamma' \vdash b : B \dashv \Delta'$
then $\Gamma; \Gamma' \vdash (a_x b) : C \dashv \Delta, \Delta'$

References

Anoun, H. and A. Lecomte. 2007. Linear grammars with labels. *Formal Grammar 2006.*

Bach, E. and B. Partee. 1980. Anaphora and semantic structure. in J. Krieman and A. Ojeda, eds., *Papers from the Parasession on Pronouns and Anaphora.* Chicago: Chicago Linguistic Society, pp. 1-28.

Bach, E. 1981. Discontinuous constituents in generalized categorial grammar. NELS 11:1-22.

Barker, C. 2002. Continuations and the nature of quantification. *Natural Language Semantics* 10:211-242.

Bernardi, R. and M. Moortgat. 2007. Continuation semantics for symmetric categorial grammar. In D. Leivant and R. de Queiroz, eds., pp. 53-71.

Blackburn, P. and J. Bos. 2005. *Representation and Inference for Natural Language.* Stanford: CSLI.

Carpenter, B. 1994. A deductive account of scope. *Proceeding of the Thirteenth West Coast Conference on Formal Linguistics*, San Diego. Stanford: CSLI.

Carpenter, B. 1997. *Type-Logical Semantics.* Cambridge, MA and London: MIT Press.

Curien, P.-L., and H. Herbelin. 2000. The duality of computation. In ICFP'00, pp. 233-243.

Church, A. 1940. A formulation of a simple theory of types. *Journal of Symbolic Logic* 5:56-68.

Cooper, R. 1975. *Montague's Semantic Theory and Transformational Syntax.* Ph.D. dissertation, University of Massachusetts at Amherst.

Cooper, R. 1983. *Quantification and Syntactic Theory.* Dordrecht: Reidel.

Danvy, O. and A. Filinski. 1990. Abstracting control. In *Proceedings of the 1990 ACM Conference on Lisp and Functional Programming.* New York: ACM Press, pp. 151-160.

Felleisen, M. 1988. The theory and practice of first-class prompts. In POPL'88. New York: ACM Press, pp. 180-190.

Fischer, M. 1972. Lambda calculus schemata. *Proceedings of the ACM Conference Proving Assertions about Programs*, pp. 104-109, SIGPLAN Notices, Vol. 7, No. 1 and SIGACT News, No. 14 (January 1972).

Gazdar, G. 1979. English as a context-free language. Unpublished manuscript dated April 1979, Cognitive Science Program, University of Sussex.

de Groote, P. 2001. Type raising, continuations, and classical logic. In R. van Rooy and M. Stokhof, eds., *Proceedings of the Thirteenth Amsterdam Colloquium*. Amsterdam: Institute for Logic, Language, and Computation, Universiteit van Amsterdam, pp. 97-101.

Gallin, D. 1975. *Intensional and Higher Order Modal Logic.* Amsterdam: North-Holland.

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

Henkin, L. 1950. Completeness in the theory of types. *Journal of Symbolic Logic* 15:81-91.

Keller, W. 1988. Nested Cooper storage: the proper treatment of quantification in ordinary noun phrases. In U. Reyle and C. Rohrer, eds., *Natural Language Parsing and Linguistic Theories*. Dordrecht: Reidel, pp. 432-447.

Leivant, D. and R. de Queiroz, eds. 2007. *Proceedings of the Fourteenth Workshop on Logic, Language, Information, and Computation (WoLLIC 2008)*. Vol. 4576 of *Lecture Notes in Computer Science*. Berlin and Heidelberg: Springer.

Montague, R. 1974. The proper treatment of quantification in ordinary English. In R. Thomason, ed., *Formal Philosophy: Selected Papers of Richard Montague*. New Haven, Yale University Press, pp. 247-270.

Moortgat, M. 1988. *Categorial Investigations: Logical and Linguistic Aspects of the Lambek Calculus.* Ph.D. dissertation, Universiteit van Amsterdam. Dordrecht: Foris.

Moortgat, M. 1996. Generalized quantifiers and discontinuous type constructors. In H. Bunt and A. van Horck, eds., *Discontinuous Constituency.* De Gruyter, pp. 181-207.

Moortgat, M. 2007. Symmetries in natural language syntax and semantics. In D. Leivant and R. de Queiroz, eds., pp. 264-268.

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

Parigot, M. 1992. $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In *Proceedings of the International Conference on Logic Programming and Automated Reasoning (LPAR)*, St. Petersburg. LNCS 624.

Parigot, M. 2000. On the computational interpretation of negation. In P. Clote and H. Schwichtenberg, eds., *Computer Science Logic*, vol. 1862 of *Lecture Notes in Computer Science*. Berlin and Heidelberg: Springer, pp. 472-484.

Plotkin, G. 1975. Call-by-name, call-by-value, and the λ-calculus. *Theoretical Computer Science* 1(2):125-159.

Pollard, C. 2007. Nonlocal dependencies via variable contexts. In R. Muskens, ed., *Workshop on New Directions in Type-Theoretic Grammar.* ESSLLI 2007, Dublin. Revised and extended version under review for a special issue of *Journal of Logic, Language, and Information.*

Pollard, C. In press. Covert movement in logical grammar. In S. Pogodalla, M. Quatrini, and C. Retoré, eds., *Logic and Grammar: Essays Presented to Alain Lecomte on the Occasion of his 60th Birthday.* LNCS/FOLLI.

Shan, C.-c. 2004. Delimited continuations in natural language: quantification and polar sensitivity. *Continuation Workshop 2004*, Venice.

Stabler, E. 1999. Remnant movement and structural complexity. In G. Bouma, E. Hinrichs, G.-J. Kruiff, and R. Oehrle, eds., *Constraints and Resources in Natural Language.* Stanford: CSLI.

Vermaat, W. 1999. *Controlling Movement: Minimalism in a Deductive Perspective.* Master's thesis, Utrecht University.