Continuations

Chris Barker
Department of Linguistics
University of California, San Diego
barker@ucsd.edu

Chung-chieh Shan
Division of Engineering and Applied Sciences, Harvard University
ccshan@post.harvard.edu

16th European Summer School in Logic, Language and Information, August 2004

Chris Barker. 2002. **Continuations and the nature of quantification.** *Natural Language Semantics* 10(3):211–242.

Chung-chieh Shan and Chris Barker. 2004. **Explaining crossover and superiority as left-to-right evaluation.** Draft manuscript.

Philippe de Groote. 2001. **Type raising, continuations, and classical logic.** In *Proceedings of the 13th Amsterdam Colloquium*, ed. Robert van Rooy and Martin Stokhof, 97–101. Institute for Logic, Language and Computation, Universiteit van Amsterdam.

Chris Barker and Chung-chieh Shan. 2004. **Types as graphs: continuations in Type Logical Grammar.** Draft manuscript.

Chung-chieh Shan. 2004. **Polarity sensitivity and evaluation order in type-logical grammar.** In *Proceedings of the 2004 human language technology conference of the North American chapter of the Association for Computational Linguistics*, vol. 2, 129–132.

Chris Barker. 2004. **Continuations in natural language.** In *Proceedings of the 4th continuations workshop*, ed. Hayo Thielecke, 55–64. Technical report CSR-04-1, School of Computer Science, University of Birmingham.

Chung-chieh Shan. 2004. **Delimited continuations in natural language: quantification and polarity sensitivity.** In *Proceedings of the 4th continuations workshop*, ed. Hayo Thielecke, 55–64. Technical report CSR-04-1, School of Computer Science, University of Birmingham.

Chris Barker. 2004. **Parasitic scope.** Abstract for talk at Semantics and Linguistic Theory 14.

CHRIS BARKER

CONTINUATIONS AND THE NATURE OF QUANTIFICATION*

This paper proposes that the meanings of some natural language expressions should be thought of as functions on their own continuations. Continuations are a well-established analytic tool in the theory of programming language semantics; in brief, a continuation is the entire default future of a computation. I show how a continuation-based grammar can unify several aspects of natural language quantification in a new way: merely stating the truth conditions for quantificational expressions in terms of continuations automatically accounts for scope displacement and scope ambiguity. To prove this claim, I exhibit a simple finite context-free grammar with a strictly compositional semantics in which quantificational NPs are interpreted in situ but take semantic scope over larger constituents. There is no Quantifier Raising (nor any use of a level of Logical Form distinct from overt syntax), no Cooper Storage (or similar mechanisms used in many recent HPSG, Categorial, or Type-logical treatments), and no need for type-shifting (as in Hendriks' Flexible Types account). Continuations also provide a natural account of generalized coordination that does not require either type-shifting or type polymorphism. Compositionality issues are discussed in some detail.

1. Introduction: The Continuation Hypothesis

This paper sets out to motivate the use of continuations as a tool for describing natural language meaning. The main result is a demonstration of how a continuation-based analysis can unify several aspects of nominal quantification in a new way. The aspects at issue correspond to the following three questions:

(1) a. **Duality of NP meaning:** What (if anything) unifies the meanings of quantificational versus non-quantificational NPs?
b. **Scope displacement:** Why does the semantic scope of a quantificational NP sometimes differ from its syntactic scope?
c. **Scope ambiguity:** How does scope ambiguity arise?

Logically, the answers to these questions could all be quite different. Of

[*] Thanks to Mark Gawron, Philippe de Groote, Ed Keenan, Chung-Chieh Shan, two anonymous referees, and audiences at UC Irvine, SALT 11, MIT, and INRIA-Lorraine. This paper was written in August 2000 and revised in August 2002; however, the revisions do not fully reflect recent advances in the understanding of continuations as applied to natural language, notably de Groote (2001), which presents an application of Parigot's λμ-calculus to quantificational issues, and Shan (2002), which proposes a type-shifting grammar based on higher-order continuations to explain superiority effects in multiple WH-questions.

course, scope ambiguity cannot exist without scope displacement; nevertheless, it makes sense to consider them separately, since on at least one influential theory of quantification (May 1985), relative scope is determined by a separate mechanism from scope displacement. In addition, the existence of scope ambiguity is open to dispute (e.g., Reinhart 1979; Chierchia and McConnell-Ginet 1990, p. 116) in a way that the manifest existence of scope displacement is not.

What do other theories have to say about the three questions in (1)? I will take Quantifier Raising at a level of Logical Form to be the dominant view of natural language quantification among linguists and perhaps among philosophers – or at the very least, the most universally familiar one. The QR story is enormously persuasive and robust, both from a descriptive and from an explanatory point of view. For the sake of concreteness, I will use Heim and Kratzer (1998) (especially chapters 6 and 7) as my reference for the standard QR view.[1] On Heim and Kratzer's version of the story, non-quantificational NPs denote entities (type e), quantificational NPs (henceforth, 'QNPs') denote generalized quantifiers (type $\langle\langle \mathsf{e}, \mathsf{t}\rangle, \mathsf{t}\rangle$), and typical transitive verbs denote relations over entities (type $\langle \mathsf{e}, \langle \mathsf{e}, \mathsf{t}\rangle\rangle$).[2] When a QNP occurs in subject position, type-driven composition allows (indeed, requires) it to take the verb phrase (type $\langle \mathsf{e}, \mathsf{t}\rangle$) as an argument. However, when a QNP occurs in a non-subject position, including direct object position, a type mismatch occurs (assuming, as Heim and Kratzer carefully note, that certain type-shifting operations are disallowed). Since interpretation would otherwise be impossible, QR adjoins the offending QNP higher in the tree, leaving behind a pronoun in the original NP position. This repairs the type mismatch, and simultaneously explains scope displacement. In addition, (in most versions of the QR approach) we also have an explanation for scope ambiguity, since we can assume that the relative scope of QNPs reflects the order in which they were raised.

Under the QR account, the best we can say in answer to the duality question in (1a) is that a generalized quantifier is what a subject NP would have to denote in order to take a verb phrase as an argument. But why are subjects special? And why repair type mismatches via QR, rather than, say, prohibiting QNPs in non-subject positions? There may be reasonable answers to these questions, perhaps along the lines of claiming that since QR resembles overt syntactic movement, we can use it "for free"; my point is that the fact that these questions require answers shows that duality and scope displacement are distinct phenomena according to the QR view.

[1] Heim and Kratzer (1998) is a textbook, and consequently the authors' discussion is simplified in ways that may weaken the coverage of their analysis compared to cutting-edge versions of QR. Besides clarity and accessibility, what makes Heim and Kratzer's version a suitable choice here is their careful development of the motivation behind the QR approach, the nature of its explanatory power, and the trade-offs in comparison with in-situ Null-LF approaches (in their case, Hendriks' Flexible Types, which is also discussed directly below). Furthermore, they provide an excellent checklist of basic (and not so basic) examples that any theory of quantification needs to explain.

[2] In Montague's (1973) type system, an expression of type e denotes an object in the set of individuals E, an expression of type t denotes a truth value, and an expression of type $\langle\alpha, \beta\rangle$ denotes a function from objects of type α to objects of type β.

Choosing the other horn of the dilemma, Montague (1973) (henceforth, PTQ) gives a compelling answer to the duality question: non-quantificational NPs and QNPs all denote generalized quantifiers. That is why they have closely similar syntactic distribution, and in the PTQ fragment, predicates accept generalized quantifiers in any NP position without type mismatch. But nothing in the type system forces QNPs to take wide scope. As a result, Montague needs to stipulate a separate operation of quantifying in to account for scope displacement and scope ambiguity. Once again we fail to provide a unified answer to the three questions in (1).

In Hendriks' (1988, 1993) Flexible Types approach (similar ideas are present in Partee and Rooth 1983 and Keenan 1987), many transitive verbs have as their basic type a relation over individuals. Hendriks provides a rule called Argument Raising that raises the type of one argument of a predicate from entity type to generalized quantifier type, and simultaneously gives that argument scope over the other arguments of the predicate. The order in which Argument Raising applies to the two arguments of, for instance, a transitive verb determines the relative scope of the arguments. Thus the single rule of Argument Raising simultaneously accounts for duality as well as at least some portion of scope displacement and scope ambiguity.

Unfortunately, the rule of Argument Raising alone is not sufficient to give a complete analysis of scope displacement and scope ambiguity. At least one additional type-shifting schema (Value Raising) is needed. Yet again, a complete unified explanation eludes us.

Therefore, consider the following proposal:

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

As explained in detail in the sections below, this single assumption provides answers to all three of the questions in (1): (a) QNPs denote the same kind of object as other types of NP, so there is no type clash when they occur in object position or other NP argument positions; (b) because of the nature of continuations, merely stating the truth conditions of a QNP in terms of continuations automatically guarantees that it will have semantic scope over an entire clause – in other words, scope displacement follows directly from the semantic nature of quantification; finally, (c) scope ambiguity turns out to be a natural consequence of indeterminacy in the way in which continuations get put together in the course of composing a complex meaning. In sum, duality, scope displacement, and scope ambiguity all follow from the single assumption that noun phrase meanings have access to their continuations.

2. Continuations

Reynolds (1993) tells how the concept of continuations emerged independently in the work of several computer scientists in the 1960s and early 1970s. In fact, with the benefit of hindsight, a limited form of continuation-passing is clearly discernible at the core of Montague's (1973) PTQ treatment of NPs as generalized quantifiers (what continuation-passing is will be explained in section 2.2, and the connection with PTQ will be developed in some detail). Currently, according to Danvy and Talcott (1998, p. 115), continuations are "ubiquitous in many different areas of computer science, including logic, constructive mathematics, programming languages, and programming."

Unfortunately, continuations are a notoriously difficult idea to explain at an intuitive level. I will make my best attempt, of course, but experience suggests that a full understanding of continuations usually emerges only after working through a number of concrete examples.

2.1. *What Is a Continuation?*

Actually, understanding what a continuation is per se is not so hard; what is more difficult is understanding how a grammar based on continuations works – what I will call a 'continuized' grammar.

Kelsey et al. (1998, p. 71) explain that "a continuation represents the entire (default) future for the computation." For instance, when computing the meaning of the sentence *John saw Mary*, the default future of the value denoted by the subject is that it is destined to have the property of seeing Mary predicated of it. In symbols, the continuation of the subject denotation **j** is the function $\lambda x.\mathbf{saw}\ \mathbf{m}\ x$.[3] Similarly, the default future of the object denotation **m** is the property of being seen by John, $\lambda y.\mathbf{saw}\ y\ \mathbf{j}$; the continuation of the transitive verb denotation **saw** is the function $\lambda R.R\ \mathbf{m}\ \mathbf{j}$; and the continuation of the VP *saw Mary* is the function $\lambda P.P\ \mathbf{j}$.

[3] When parentheses are omitted, functional application is left-associative, as usual, so that $\mathbf{saw}\ \mathbf{m}x = ((\mathbf{saw}\ \mathbf{m}))(x))$.

This simple example illustrates two important aspects of continuations: (1) every meaningful subexpression has a continuation; and (2) the continuation of an expression is always relative to some larger expression containing it. Thus when *John* occurs in the sentence *John left yesterday*, its continuation is the property $\lambda x.\mathbf{yesterday}\ \mathbf{left}\ x$; when it occurs in *Mary thought John left*, its continuation is the property $\lambda x.\mathbf{thought}\ (\mathbf{left}\ x)\ \mathbf{m}$; when it occurs in the sentence *Mary or John left*, its continuation is $\lambda x.(\mathbf{left}\ \mathbf{m}) \vee (\mathbf{left}\ x)$; and so on.

For those readers familiar with Rooth's alternative-set semantics for focus constructions, it may be helpful to think of a continuation as roughly similar in form to the abstract that generates the focus alternative set when the expression in question is in focus.

In general, given a subexpression A of type α embedded in an expression B of type β, the continuation of A relative to B is a function of type $\langle\alpha, \beta\rangle$ abstracting over the denotation of A that characterizes how the meaning of the whole depends on the meaning of the subexpression. Continuations, then, are just a different way of looking at the relationship between the meaning of a complex expression and the meaning of its parts.

Clearly, continuations exist independently of any framework or specific analysis, and all occurrences of expressions have continuations in any language that has a semantics. Since continuations are nothing more than a perspective, they are present whether we attend to them or not. The question under consideration, then, is not whether continuations exist – they undoubtedly do – but precisely how natural language expressions do or don't interact with them.

2.2. *Continuizing a Simple Grammar: Deriving Generalized Quantifiers as a Special Case*

The basic idea of continuizing a grammar is to provide subexpressions with direct access to their continuations. In order to do this, the denotations of the subexpressions must be modified to take a continuation as an argument.

In the theoretical computer science literature, a program modified in this way is said to be written in CONTINUATION-PASSING STYLE, and the relationship between a non-continuized program and a continuized one is given as a syntactic transformation called a CPS TRANSFORM. Since we are discussing grammars for natural language, the relevant transform here takes a non-continuized grammar and produces the equivalent continuized grammar. I will say that a grammar before continuization has a DIRECT semantics, and a grammar after continuization has a CONTINUIZED semantics.

I will build a continuized fragment with quantification in three main steps: first, I will present a non-quantificational grammar with a direct semantics. Then I will construct an equivalent continuized grammar. Finally, I will add elements to the continuized grammar that exploit the presence of continuations in order to account for scope displacement and scope ambiguity.

First, the direct semantics. Consider the following simple context-free syntax with an extensional semantics for a fragment without quantification.

(3)	SYNTAX	DIRECT SEMANTICS
	S → NP VP	⟦VP⟧(⟦NP⟧)
	VP → Vt NP	⟦Vt⟧(⟦NP⟧)
	NP → John	**j**
	NP → Mary	**m**
	VP → left	**left**
	Vt → saw	**saw**

Leaving the model-theoretic interpretation of the logical language implicit (but, I trust, obvious), we have the following translations after lambda conversion:

(4)	a.	John left.	**left j**
	b.	John saw Mary.	**saw m j**

Note that this grammar operates with the following types for each syntactic category it recognizes:

(5)	DIRECT	SEMANTIC	
	DENOTATION	TYPE	DESCRIPTION
	⟦S⟧	t	truth value
	⟦NP⟧	e	entity
	⟦VP⟧	⟨e, t⟩	property (i.e., a set of entities)
	⟦Vt⟧	⟨e, ⟨e, t⟩⟩	relation over entities

This grammar, needless to say, will not accommodate quantificational NPs.

It will be helpful to be similarly explicit about the semantic types of some of the symbols used in the logical translation language.

(6)	DIRECT VARIABLE	TYPE
	p, q	t
	x, y, z	e
	P, Q	⟨e, t⟩
	R, S	⟨e, ⟨e, t⟩⟩

Logical constants such as **j**, **left**, and **saw** have the semantic type of their corresponding syntactic category (for these examples, e, ⟨e, t⟩, and ⟨e, ⟨e, t⟩⟩, respectively).

Since we will be dealing in continuations, we will need new logical symbols to use as variables over continuation objects. Somewhat abusing the notational conventions of computer science discussions of continuations, I will relate a continuation variable to its direct counterpart by drawing a line over the continuation variable:

(7)	CONTINUATION VARIABLE	TYPE
	$\overline{p}$, $\overline{q}$	⟨t, t⟩
	$\overline{x}$, $\overline{y}$, $\overline{z}$	⟨e, t⟩
	$\overline{P}$, $\overline{Q}$	⟨⟨e, t⟩, t⟩
	$\overline{R}$, $\overline{S}$	⟨⟨e, ⟨e, t⟩⟩, t⟩

Comparing (6) with (7), the relationship between a direct variable and its continuation counterpart should be fairly transparent. Since continuations are functions from the type of the direct value to the type of the meaning of the larger expression as a whole, and since in this paper the larger expression will always be a declarative sentence of type t, the type of the continuation variable will be a function from the type of the corresponding direct variable to a truth value. For instance, since P is a variable of type ⟨e, t⟩, $\overline{P}$ is a variable of type ⟨⟨e, t⟩, t⟩.

The goal of continuizing the grammar is to provide each expression with its own continuation as an argument. Therefore if an expression of category A with semantic type α has direct denotation ⟦A⟧, then the continuized denotation ⦃A⦄ will be a function from A-continuations (type ⟨α, t⟩) to the type of the larger expression as a whole, i.e., to a truth value: type (⦃A⦄) = ⟨⟨α, t⟩, t⟩.

(8)	CONTINUIZED DENOTATION	TYPE
	⦃S⦄	⟨⟨t, t⟩, t⟩
	⦃VP⦄	⟨⟨⟨e, t⟩, t⟩, t⟩
	⦃NP⦄	⟨⟨e, t⟩, t⟩
	⦃Vt⦄	⟨⟨⟨e, ⟨e, t⟩⟩, t⟩, t⟩

For instance, a continuized VP is a function from VP continuations to truth values.

In section 5, I will give a general procedure for continuizing an arbitrary compositional grammar in which the nature of continuization is captured in a single parameterized schema. Because of the high level of generality involved, the definitions with the parameters are rather abstract; therefore at this stage in the exposition I will give several separate schemata, and show in section 5 how they correspond to instances of the general schema. To begin, we will need one rule for lexical entries and one for functional application:

(9)	Syntax	Direct Semantics	Continuized Semantics
	A → B	$[\![\mathrm{B}]\!]$	$\lambda\overline{b}.\overline{b}(\{\!\!\{\mathrm{B}\}\!\!\})$
	A → B C	$[\![\mathrm{B}]\!]([\![\mathrm{C}]\!])$	$\lambda\overline{a}.\{\!\!\{\mathrm{B}\}\!\!\}(\lambda b.\{\!\!\{\mathrm{C}\}\!\!\}(\lambda c.\overline{a}(bc)))$

Here *a*, *b*, and *c* are logical variables of the same type as the direct values of A, B, and C, respectively. As desired, each syntactic constituent takes a continuation as an argument.

Continuations as implemented here are significantly simpler than standard programming language treatments of continuations. The opportunity for simplicity comes from the fact that none of our composition rules involve creating a new lambda abstract. In particular, there will be no need to posit a rule corresponding to predicate abstraction, as there is in the treatment of Quantifier Raising in Heim and Kratzer (1998, p. 186).

Applying the schemata in (9) to the direct grammar in (3) gives a continuized grammar:

(10)	Syntax	Continuized Semantics
a.	S → NP VP	$\lambda\overline{p}.\{\!\!\{\mathrm{VP}\}\!\!\}(\lambda P.\{\!\!\{\mathrm{NP}\}\!\!\}(\lambda x.\overline{p}(Px)))$
b.	VP → Vt NP	$\lambda\overline{P}.\{\!\!\{\mathrm{Vt}\}\!\!\}(\lambda R.\{\!\!\{\mathrm{NP}\}\!\!\}(\lambda x.\overline{P}(Rx)))$
c.	NP → John	$\lambda\overline{x}.\overline{x}(\mathbf{j})$
d.	NP → Mary	$\lambda\overline{x}.\overline{x}(\mathbf{m})$
e.	VP → left	$\lambda\overline{P}.\overline{P}(\mathbf{left})$
f.	Vt → saw	$\lambda\overline{R}.\overline{R}(\mathbf{saw})$

An example will show how the continuized grammar computes a result equivalent to the basic grammar. First, note that the syntax of the continuized grammar in (10) is identical to that of the original grammar in (3).

(11)		
	[S [NP John] [VP left]]	Syntax for *John left*
	$\lambda\overline{p}.\{\!\!\{\mathrm{left}\}\!\!\}(\lambda P.\{\!\!\{\mathrm{John}\}\!\!\}(\lambda x.\overline{p}(Px)))$	Rule (10a)
	$\lambda\overline{p}.\{\!\!\{\mathrm{left}\}\!\!\}(\lambda P.((\lambda\overline{x}.\overline{x}(\mathbf{j}))(\lambda x.\overline{p}(Px))))$	Rule (10c)
	$\lambda\overline{p}.\{\!\!\{\mathrm{left}\}\!\!\}(\lambda P.\overline{p}(P\ \mathbf{j}))$	β-conversion
	$\lambda\overline{p}.(\lambda\overline{P}.\overline{P}(\mathbf{left}))(\lambda P.\overline{p}(P\ \mathbf{j}))$	Rule (10e)
	$\lambda\overline{p}.\overline{p}(\mathbf{left\ j})$	β-conversion

In the continuized grammar, *John left* denotes a function from sentence continuations to a truth value (type ⟨⟨t, t⟩, t⟩). If we provide $\{\!\!\{\textit{John left}\}\!\!\}$ with the most trivial continuation possible (namely, the identity function, $\lambda p.p$), we get (12).

(12)		
	$(\lambda\overline{p}.\overline{p}(\mathbf{left\ j}))(\lambda p.p)$	application to the trivial continuation
	$\mathbf{left\ j}$	β-conversion

It is easy to verify for this simple case that, modulo application to the trivial continuation, the continuized grammar always computes the same values as the original direct grammar. This equivalence is proven for the general case in section 5.

Note in (8) that a continuized NP denotation is of type ⟨⟨e, t⟩, t⟩. That, of course, is the type of (an extensional) generalized quantifier as proposed in PTQ. This is the sense in which PTQ involves a limited form of continuation-passing: saying that NPs denote generalized quantifiers amounts to saying that NPs denote functions on their own continuations.

This result bears emphasizing: the generalized quantifier conception of NP meaning falls out automatically from continuizing a non-quantificational grammar. Moreover, unlike the treatment in PTQ, nothing in the continuization transform is NP specific. In other words, the treatment of NPs as generalized quantifiers is a special case of the more general concept of continuization.

2.3. *The Nature of Quantification*

So far all we have done is construct a continuized grammar that is equivalent to the original basic grammar. We are now in a position to provide truth conditions for some quantificational expressions.

(13)		
a.	NP → everyone	$\lambda\overline{x}\forall x.\overline{x}(x)$
b.	NP → someone	$\lambda\overline{x}\exists x.\overline{x}(x)$

These rules have no counterpart in the direct grammar. This amounts to saying that the meanings of *everyone* and *someone* are essentially quantificational; their meanings can be expressed only in terms of an already-continuized grammar like that in (10).

Note that the denotation of the NP *everyone* is the same type as the denotation of a continuized NP in (10), namely, a function from NP continuations to truth values (type $\langle\langle \mathsf{e}, \mathsf{t}\rangle, \mathsf{t}\rangle$). (And in general, all members of a given syntactic category denote objects of the same semantic type.) This is the answer to the question in (1a) concerning the duality of NP meaning: QNPs and other NPs denote the same kind of semantic object, which accounts for their syntactic and semantic interchangeability. QNP denotations differ from the denotations of other NPs only in that QNPs take advantage of the presence of continuations in a way that non-quantificational NPs do not.

More specifically, the rule in (13a) says that when *everyone* is used in a context in which $\overline{x}$ is its continuation, the semantic result depends on trying all the possible individuals that might be fed to that continuation. Similarly, the denotation of *someone* takes its continuation and wraps an existential quantification around it.

An example will show how these rules work. When the rules in (13) are added to the continuized grammar in (10), *Everyone left* smoothly evaluates to $\forall x.\mathbf{left}\ x$. Unlike the QR treatment, however, when a QNP occurs in direct object position, there is no type clash, and the computation proceeds just as smoothly:

(14) John saw everyone.
[S [NP John] [VP [Vt saw] [NP everyone]]]
$\lambda\overline{p}.\llbracket saw\rrbracket(\lambda R.\llbracket everyone\rrbracket(\lambda y.\llbracket John\rrbracket(\lambda x.\overline{p}(Ryx))))$
$\lambda\overline{p}.\llbracket everyone\rrbracket(\lambda y.\llbracket John\rrbracket(\lambda x.\overline{p}(\mathbf{saw}\ yx)))$
† $\lambda\overline{p}.\llbracket everyone\rrbracket(\lambda y.\overline{p}(\mathbf{saw}\ y\ \mathbf{j}))$
$\lambda\overline{p}.((\lambda\overline{x}\forall x.\overline{x}x))(\lambda y.\overline{p}(\mathbf{saw}\ y\ \mathbf{j})))$
$\lambda\overline{p}.\forall x.\overline{p}(\mathbf{saw}\ x\ \mathbf{j})$

Applying this denotation to the trivial continuation, we get $\forall x.\mathbf{saw}\ x\ \mathbf{j}$, which is a reasonable (extensional) denotation for the sentence *John saw everyone*.

The line marked with a dagger ('†') reveals that the continuation for the direct object NP *everyone* is $\lambda y.\overline{p}(\mathbf{saw}\ y\ \mathbf{j})$: roughly (ignoring the continuation variable $\overline{p}$) the property of being seen by John. Unlike what we find in Quantifier Raising analyses, this property does not correspond to any syntactic or logical constituent; rather, the daggered line is guaranteed to be semantically equivalent to the meaning of the sentence as a whole by beta-reduction.

In any case, there is no type clash or asymmetry between quantificational NPs occurring in subject and non-subject positions, as there is in the QR account.

Quantificational determiners. The rules in (13) treat *everyone* and *someone* as lexical (i.e., syntactically unanalyzed) NPs. Most QNPs, of course, are syntactically complex and contain a quantificational determiner. The Appendix explains how the continuation analysis naturally leads to considering all determiners – even quantificational ones – as having denotations based on choice functions. The result looks very different from the traditional generalized quantifier treatment as in, e.g., Barwise and Cooper (1981). Trying to understand how to continuize a grammar and at the same time to also re-analyze determiners as choice functions is a heavy load; therefore, for purely expository reasons, for now I will provide a special composition rule (i.e., a composition rule that is not an instance of the schemata in (9)) for quantificational determiners:

(15)	Syntax	Semantics
	NP → Det N	$\llbracket\mathrm{Det}\rrbracket(\llbracket\mathrm{N}\rrbracket)$

This allows for (comparatively) familiar lexical entries for quantificational determiners along the following lines:

(16)	every	$\lambda\overline{P}\overline{x}.\overline{P}(\lambda P\,\forall x.Px \rightarrow \overline{x}(x))$
	a, some	$\lambda\overline{P}\overline{x}.\overline{P}(\lambda P\,\exists x.Px \wedge \overline{x}(x))$
	most	$\lambda\overline{P}\overline{x}.\overline{P}(\lambda P.\mathbf{most}(P)(\overline{x}))$
	no	$\lambda\overline{P}\overline{x}.\overline{P}(\lambda P.\neg\exists x.Px \wedge \overline{x}(x))$

Here, →, ∧, and ¬ are the standard logical connectives defined over truth values, and **most** is the familiar relation over sets used in, e.g., Barwise and Cooper (1981). Recall that $\overline{x}$ is a variable over NP continuations, and therefore is a function from individuals to truth values, which is the same type as the direct variable P. Thus these denotations take a continuized nominal denotation as an argument and return a continuized NP meaning.

A few examples will illustrate these definitions in action. The grammar consisting of the union of the rules in (10), (13), (15), and (16) generates the following analyses:

(17) a. John saw every men. $\forall x.\mathbf{man}\ x \rightarrow \mathbf{saw}\ x\ \mathbf{j}$
b. John saw most men. $\mathbf{most}(\mathbf{man})(\lambda x.\mathbf{saw}\ x\ \mathbf{j})$
c. Every man saw a woman. $\exists y.\mathbf{woman}\ y \wedge \forall x.\mathbf{man}\ x \rightarrow \mathbf{saw}\ yx$

Note that the interpretation in (17c) corresponds to inverse scope, since the direct object takes scope over the subject. This shows that despite being an ‘in situ’ analysis, nothing in the continuation mechanism itself biases towards linear scope or inverse scope. (Arriving at the opposite scoping is the topic of the next subsection.)

In sum, continuations allow NPs to function as generalized quantifiers in any syntactic argument position. Furthermore, stating the truth conditions for quantificational NPs in terms of continuations automatically accounts for scope displacement.

2.4. *Scope Ambiguity: A Question of Priority*

The analysis so far provides reasonable interpretations for sentences involving quantifiers, but it provides only one interpretation for each sentence. How does relative scope ambiguity arise?

The answer comes from the fact that there is more than one way to continuize a given composition rule. The continuized grammar given in (10) contains rule (18a):

(18) a. S → NP VP $\lambda\overline{p}.\{\!\{\mathrm{VP}\}\!\}(\lambda P.\{\!\{\mathrm{NP}\}\!\}(\lambda x.\overline{p}(Px)))$
b. S → NP VP $\lambda\overline{p}.\{\!\{\mathrm{NP}\}\!\}(\lambda x.\{\!\{\mathrm{VP}\}\!\}(\lambda P.\overline{p}(Px)))$

But we might just as well have used (18b). Substituting (18b) in the example grammar will allow the subject to take wide scope over the VP. (Both (18a) and (18b) are instances of the general continuization schema discussed in section 5.)

How shall we interpret this state of affairs? Given the equation S = VP(NP), we can either interpret the NP as providing the continuation for the VP (“What you do with a VP is apply it as a functor to the subject”) or we can interpret the VP as providing the continuation for the subject (“What you do with a subject is feed it as an argument to a VP”). The result is the same, in the absence of quantification – but in the presence of quantification, the two perspectives lead to different relative scopings.

Computationally, the two rules in (18) correspond to different orders of execution. In fact, one of the main applications of continuations to date (see, e.g., Meyer and Wand 1985, p. 223) is to model programming languages in which evaluating expressions may have side effects, in which case the behavior of the program may differ depending on whether arguments are evaluated left-to-right or right-to-left. If left-to-right order of evaluation is desired, only rules like (18a) are included in the continuized grammar, and vice versa for right-to-left evaluation. In a grammar modeling a natural language, of course, we can have both types of rules, leading to ambiguity.

Therefore let us say that (18a) gives the VP PRIORITY over the NP, so

that quantificational elements in the VP take scope over the NP. Similarly, (18b) gives the NP priority over the VP, so that the subject takes wide scope.

Since both prioritizations are equally valid ways of providing access to continuations, unless we say something extra, both are equally available for use. Thus stating the meaning of quantificational elements by means of continuations automatically predicts not only scope displacement, but scope ambiguity as well.

2.5. *Bounding Scope Displacement*

In general, scope displacement can cross an unbounded number of syntactic levels.

(19) a. A raindrop fell on every car.
b. A raindrop fell on the hood of every car.
c. A raindrop fell on the top of the hood of every car.

It is easy to see how to extend this series ad infinitum. The most natural reading of these sentences requires that *every* take wide scope over *a raindrop*. (See the Appendix for the complete fragment that generates examples like (19).)

However, some people believe that QNPs cannot take scope outside of their minimal tensed S. If so, then something special must be said about tensed Ss (just as in every other theory of quantifier scope). One way to accomplish this here is to adjust the composition rules for the S node so as to disrupt the transmission of continuation information between the subconstituents and the S:

(20) a. OLD: S → NP VP $\lambda\overline{p}.\{\!\{\mathrm{VP}\}\!\}(\lambda P.\{\!\{\mathrm{NP}\}\!\}(\lambda x.\overline{p}(Px)))$
b. NEW: S → NP VP $\lambda\overline{p}.\overline{p}(\{\!\{\mathrm{VP}\}\!\}(\lambda P.\{\!\{\mathrm{NP}\}\!\}(\lambda x.Px)))$

(A similar adjustment needs to be made in the S rule given in (18b) with the opposite scoping priority; see the fragment in the Appendix.) The difference is that the occurrence of the clause’s continuation, $\overline{p}$, is inside the scope of the subject and of the verb phrase in (20a), but is outside in (20b).

(21) a. A man thought everyone saw Mary.
b. $\exists y.\mathbf{man}\ y \wedge \mathbf{thought}(\forall x.\mathbf{saw}\ \mathbf{m}\ x)\ y$

Given the revision in (20b), all scopings of (21a) are logically equivalent to (21b). That is, *every* is not able to take scope outside of the embedded clause.

Note that the adjustment in (20b) can only be made for syntactic categories whose direct (i.e., uncontinuized) type is t, since the value returned by the outermost continuized function must serve as the argument to the continuation for that expression. (See Heim and Kratzer (1998, p. 215) for a derivation of the analogous constraint in QR theories.)

2.6. *Summary of Section 2*

At this point we have a unified explanation for the questions in (1). As for the puzzle of NP Duality, QNPs denote the same type of function as other NPs, which explains their syntactic interchangeability; quantificational and non-quantificational NPs differ only in that QNPs exploit the presence of continuations in a way that non-quantificational NPs do not. As for scope displacement and scope ambiguity, merely stating the truth conditions for QNPs in terms of continuations automatically accounts for scope displacement and scope ambiguity without further stipulation.

3. NP as a Scope Island

One advantage of the account of scoping given in section 2 is that it automatically provides both linear and inverse scope when a QNP is embedded within an NP:

(22) a. [No man from a foreign country] was admitted.
b. $\neg\exists x \exists y.\mathbf{man}\ x \wedge \mathbf{country}\ y \wedge \mathbf{from}\ y\ x \wedge \mathbf{admitted}\ x$
c. $\exists y \neg\exists x.\mathbf{man}\ x \wedge \mathbf{country}\ y \wedge \mathbf{from}\ y\ x \wedge \mathbf{admitted}\ x$

The QNP *a foreign country* is embedded within the subject NP. On the most prominent reading of (22a), *no* takes scope over *a*, resulting in the truth conditions given in (22b). On the continuation analysis, this reading is produced automatically when the quantificational determiner *a* is given priority over the nominal *from a foreign country*. There is also a so-called inverse linking reading, as given in (22b), which arises when the nominal is given priority over the determiner. (The inverse linking reading is more prominent in other examples, such as *An apple in every basket was rotten*.)

Linear scope in examples like (22) poses a problem for QR theories. If QR simply adjoins QNPs to S, the obvious thing to try is to first raise *a foreign country* to S, then raise *No student from t*, where *t* is the trace left by QR of *a foreign country*. This gives the correct scoping, but leaves the trace of the first raised NP unbound, leading to an uninterpretable structure, or at least to incorrect truth conditions.

May (1985) and Richard Larson (in unpublished work described in Heim and Kratzer 1998, p. 233) conclude that NP is a scope island: an NP embedded within an NP may move only as far as its containing NP. It is fairly natural to decide that NP is a scope island, since NP is an island for overt syntactic movement, and QR is generally supposed to obey the same constraints that govern syntactic movement. However, allowing a QNP to adjoin to NP creates an awkward problem for interpretation: it requires type flexibility, since the raised NP is not the right type when adjoined within NP to receive its normal interpretation (see Heim and Kratzer 1998, p. 221). Therefore it is to the credit of the continuation analysis that it handles linear and inverse scope for QNPs within NP without special stipulation.

Interestingly, in addition to providing linear scope and preventing unbound traces from arising, making NP a scope island makes good predictions with respect to the observed range of scope orderings, as May and Larson note.

(23) a. Two politicians spy on [someone from every city]. (May)
b.*every city > two politicians > someone

May observes that sentence (23a) does not have the scoping indicated in (23b). This is explained if QR must adjoin to the closest containing NP, since that would prevent *every city* from escaping from the object NP; as a result, there is no way that *two politicians* can take scope over *someone* without also taking scope over *every city*.

The continuation analysis also predicts the impossibility of the scoping in (23). In fact, the continuation fragment obeys a more general constraint that I call the SYNTACTIC CONSTITUENT INTEGRITY scoping constraint (Barker 2001). Integrity requires that if there is a syntactic constituent that contains B and C but not A, then A must take scope over both B and C or neither. The reason the continuation analysis obeys Integrity has to do with the nature of the priority relation:

(24) $[\ldots A \ldots]_X\ [\ldots B \ldots C \ldots]_Y$

Either X (and everything X contains) will take priority over Y (and take scope over everything within Y), or else Y (and everything within Y) will take scope over X. Thus (23) is a special case of Integrity:

(25) [Two politicians$_\mathrm{A}$] [spy on someone$_\mathrm{B}$ from every city$_\mathrm{C}$]

The Integrity constraint says that *two politicians* cannot take scope over *someone* without also taking scope over *every city*, correctly predicting that the scoping in (23b) is impossible. Once again, the continuation analysis makes a good prediction without needing to stipulate any special property of NPs.

It is worth mentioning that Integrity also limits possible scopings when a ditransitive verb has three quantificational NPs as arguments, since only four of the six possible factorial scopings are consistent with Integrity:

(26) a. Most subjects put an object in every box.
b.**every box* > *most subjects* > *an object*

For instance (assuming that *put an object in every box* is a syntactic constituent on every syntactic analysis of (26a)), Integrity predicts that (26a) cannot have the scoping in (26b). In Barker (2001), I argue that this is a good prediction empirically; however, the issue is fairly intricate, since it is necessary to factor out specific indefinites, syntactic ambiguity, collective and distributive readings, and function composition of the sort proposed in Steedman (2000). See Barker (2001) for discussion.

In sum, QR without constraints creates unbound traces and incorrect truth conditions. Making NP a scope island prevents unbound traces and accounts for one kind of Integrity effect; however, it also creates a type mismatch. The continuation analysis, remarkably, with no special stipulations at all gets linear scoping, inverse scoping, and NP-related Integrity effects right.

4. Generalized Coordination without Type-Shifting or Type Polymorphism

The question naturally arises whether other linguistic elements manipulate continuations besides quantificational NPs. This section shows how continuations can provide an account of generalized conjunction that is simpler than other accounts in certain specific ways.

(27) a. John left and John slept. **and**(**left j**)(**slept j**)
b. John left and slept. **and**(**left j**)(**slept j**)
c. John saw and liked Mary. **and**(**saw m j**)(**liked m j**)
d. John and Mary left. **and**(**left j**)(**left m**)

The examples in (27) illustrate coordination of S, VP, Vt, and NP. As the translations indicate (see the Appendix for details), the truth conditions of one reading of each of these sentences can be accurately expressed by unpacking the coordination into conjoined clauses. Of course, there are are a variety of other uses of *and* that cannot be paraphrased by means of conjoined clauses (*John and Mary are a happy couple*; *the flag is red and white*) that I will not discuss (see the more complete disclaimer in Partee and Rooth 1983, p. 361). Disjunction, unlike conjunction, seems to have only a generalized use, i.e., it is always possible to find a paraphrase involving disjoined clauses; in any case, it is generalized coordination that will interest us here.

I will call the sense of *and* whose meaning is equivalent to conjoined sentences GENERALIZED *and*. The question, then, is how best to capture what the different uses of generalized *and* in (27) (similarly, for *or*) have in common semantically.

Partee and Rooth (1983), building on work of Gerald Gazdar, Arnim von Stechow, and especially of Ed Keenan and Aryeh Faltz, give an analysis of generalized coordination that has three parts. First, they stipulate that a CONJOINABLE TYPE is any type ending in t. Examples include sentences (type t), verb phrases and common nouns (type ⟨e, t⟩), and quantificational NPs (type ⟨⟨e, t⟩, t⟩), but not the basic (i.e., lexical) type of proper names (type e).

Second, they rely on a syntactic schema to generalize over the conjoinable syntactic categories.

(28) Syntax Semantics
$X \rightarrow X_l$ and X_r $\mathbf{and}_{\langle\alpha,\,\beta\rangle}\,([\![X_l]\!])([\![X_r]\!])$

Third, they provide a recursive rule characterizing how the meaning of generalized *and* for complex semantic types relates to the meaning of *and* for simpler types. Let *L* and *R* be meanings of type ⟨α, β⟩. Then Partee and Rooth (1983) have:

(29) $\mathbf{and}_{\langle\alpha,\,\beta\rangle}\,(L)(R) = \lambda a.\mathbf{and}_{\beta}(L(a))(R(a))$

where *a* is a variable over objects of type α. The base case says that $\mathbf{and}_{t}$ is the standard binary operator over truth values.

On this analysis generalized *and* has a single meaning, but that meaning is type polymorphic (i.e., able to take arguments of different semantic types). For instance, if we instantiate the syntactic schema for coordinating VPs, then the denotation of *and* takes properties (here, type ⟨e, t⟩) as arguments; but if we instantiate the syntactic schema for coordinating transitive verbs, then the denotation of *and* takes relations as arguments (type ⟨e, ⟨e, t⟩⟩).

Equivalently, it is possible to think of (29) as a type-shifting rule, in which case *and* is polysemous, where each distinct homophonous version of *and* takes arguments of a single semantic type. Then there is one basic lexical meaning for *and* (namely, the operator over truth values), and the various other senses of *and* are related to each other and ultimately to the basic lexical meaning by means of a type-shifting rule resembling (29). See Heim and Kratzer (1998, p. 182) for a discussion of this type of approach.

On either construal (type polymorphic versus type-shifting), the claim is that *and* has a meaning that is capable of relating properties, or relations, or any other semantic objects with a conjoinable type, in addition to truth values.

Now consider one way of achieving a similar analysis of generalized coordination in a continuized grammar. Let X be a syntactic category whose direct semantic type is x.

(30) SYNTAX SEMANTICS

$X \to X_l$ and X_r $\lambda\overline{x}.\mathbf{and}(\{\!\{X_l\}\!\}(\overline{x}))(\{\!\{X_r\}\!\}(\overline{x}))$

The meaning of *and* distributes the continuation belonging to the coordinate structure across the conjuncts. Recalling that a continuation is the default future of a computation, we can gloss this rule as saying, “Whatever you are planning to do with the value of the coordinate structure, do it to the left conjunct, also do it to the right conjunct, and conjoin the resulting truth values.”

Just like (28), (30) schematizes over a range of conjoinable syntactic categories. However, there is no need to state a separate recursive rule characterizing the function denoted by the conjunction – the semantic rule in (30) gives the desired result automatically. It mentions only the basic truth-value operator **and**, and there is no need to construct semantic operators that take arguments having complex types.

Furthermore, there is no need to stipulate what counts as a conjoinable type. The result of applying any continuized denotation to a continuation (e.g., “$\{\!\{X_l\}\!\}(\overline{x})$”) is guaranteed to be a truth value, by construction. In other words, the notion of a conjoinable type is built into the structure of the continuation system. In the present context, we can restate this as follows: the observation that conjoinable types are those types that “end in t” follows from the claim that generalized coordination lives at the level of continuations.

In sum, Partee and Rooth (1983) need a syntactic schema, a recursive semantic definition, and a notion of conjoinable type. But if expressions are allowed to manipulate their continuations, all that is needed is the single syntactic schema in (30).

It might seem as if having a syntactic schema like (30) amounts to the same kind of polymorphism as the Partee and Rooth account. But there is a significant difference: Partee and Rooth have both syntactic polymorphism (represented in rule (28)) and also lexical semantic type polymorphism (as in rule (29), or, equivalently, using semantic type-shifting). On the continuations account, there is no need to recapitulate the syntactic polymorphism in the semantics. As a result, the model-theoretic meaning attributed to *and* is simpler on the continuations account: it has a single type (namely, $\langle\langle t, t\rangle, t\rangle$), and operates on truth values and nothing else.

Furthermore, the opportunity for this optimally simple meaning for generalized *and* comes directly from continuizing the grammar. This is because the recursive unpacking of the pointwise definition of Partee and Rooth’s generalized *and* (as expressed by (29)) is built into the continuation mechanism itself. To the extent that continuations are necessary independently to handle quantification, we get a simple account of the meaning of generalized coordination for free.

5. THE CONTINUATION SCHEMA

As mentioned in section 2, the relationship between a direct grammar and the equivalent continuized grammar can be expressed in a single schema. This is important for making precise the sense in which continuization constitutes a coherent, unitary operation.

The schema will show how to take a direct grammar G and produce an equivalent continuized grammar $\overline{G}$. Much of the complexity in stating the schema will come from making it as general as possible. Literally any grammar with a compositional semantics can be continuized – including, incidentally, a grammar that has already been continuized, leading to higher-order continuations.

Let G be a grammar for which each rule r has the following form:

$$\langle E^r(e_1, \ldots, e_n), C^r(c_1, \ldots, c_n), M^r(m_1, \ldots, m_n)\rangle$$

We say that rule r has arity n. The interpretation is as follows: if e_i is a well-formed expression of category c_i with meaning m_i for $0 < i \leq n$, then $E^r(e_1, \ldots, e_n)$ is a well-formed expression of category $C^r(c_1, \ldots, c_n)$ with meaning $M^r(m_1, \ldots, m_n)$.

In the context-free grammars used in this paper, the syntactic formation operation E^r is always concatenation, and the meaning composition function M^r is always functional application, but other syntactic and semantic operations are possible. For instance, for some r, E^r could be head wrap, quantifying in, QR, etc.

Then for each rule r in G with arity n, the continuized grammar $\overline{G}$ will have $n!$ corresponding rules, one for each distinct permutation function f, where f is an automorphism on the first n ordinals. For each such f, there will be a rule in $\overline{G}$ of the following form:

$$\langle E^r(e_1, \ldots, e_n), C^r(c_1, \ldots, c_n), M^r_f(\hat{m}_1, \ldots, \hat{m}_n)\rangle$$

where M^r_f satisfies the following constraint:

(31) **The Continuation Schema:** $M^r_f(\hat{m}_1, \ldots, \hat{m}_n) =$
$\lambda\overline{x}.\hat{m}_{f(1)}(\lambda x_{f(1)}[\ldots[\hat{m}_{f(n)}(\lambda x_{f(n)}[\overline{x}(M^r(x_1, \ldots, x_n))])]\ldots])$

(Here and throughout the paper I use square brackets instead of parentheses purely as a visual aid to clarity.) The hats over some semantic values ('$\hat{m}$') indicate that they are continuized meanings that take a continuation as their only argument. We shall see that the permutation functions f determine the scoping priority of the subconstituents.

It will be helpful to examine some specific instances of the schema, in order to illustrate its behavior, and also to show that the schemata given earlier actually are instances of the general schema, as promised. In addition, the structure of the proof presented later in this section will closely follow the examples given here.

For rules of arity $n = 0$ (i.e., a lexical entry), $n! = 1$, in which case the only permutation function f is undefined for every integer, and the following correspondence satisfies the Continuation Schema:

(32) Rule in G: $\langle E^r, C^r, M^r \rangle$
Rule in $\overline{G}$: $\langle E^r, C^r, \lambda\overline{x}.\overline{x}(M^r) \rangle$

This is equivalent to the schema for lexical items given in (9).

For rules of arity $n = 1$ (unary productions), $n! = 1$, and the only permutation function f maps 1 onto 1 and is undefined otherwise:

(33) Rule in G: $\langle E^r(e_1), C^r(c_1), M^r(m_1) \rangle$
Rule in $\overline{G}$: $\langle E^r(e_1), C^r(c_1), \lambda\overline{x}.\hat{m}_1(\lambda x_1.\overline{x}(M^r(x_1))) \rangle$

There is one unary rule in the fragment in the Appendix (the PP rule involving semantically transparent prepositions).

For rules of arity $n = 2$, $n! = 2$, and there are two permutation functions, f and g:

(34) Rule in G: $\langle E^r(e_1, e_2), C^r(c_1, c_2), M^r(m_1, m_2) \rangle$
Rules in $\overline{G}$: $\langle E^r(e_1, e_2), C^r(c_1, c_2), M_f^r(\hat{m}_1, \hat{m}_2) \rangle$
$\langle E^r(e_1, e_2), C^r(c_1, c_2), M_g^r(\hat{m}_1, \hat{m}_2) \rangle$
where
$M_f^r(\hat{m}_1, \hat{m}_2) = \lambda\overline{x}.\hat{m}_{f(1)}(\lambda x_{f(1)}.\hat{m}_{f(2)}(\lambda x_{f(2)}.\overline{x}(M^r(x_1, x_2))))$
$M_g^r(\hat{m}_1, \hat{m}_2) = \lambda\overline{x}.\hat{m}_{g(1)}(\lambda x_{g(1)}.\hat{m}_{g(2)}(\lambda x_{g(2)}.\overline{x}(M^r(x_1, x_2))))$

Assuming that f maps 1 to 1 and 2 to 2, and g maps 1 to 2 and 2 to 1, the rules in (34) are equivalent to those in (35):

(35) Rules in $\overline{G}$:
$\langle E^r(e_1, e_2), C^r(c_1, c_2), \lambda\overline{x}.\hat{m}_1(\lambda x_1.\hat{m}_2(\lambda x_2.\overline{x}(M^r(x_1, x_2)))) \rangle$
$\langle E^r(e_1, e_2), C^r(c_1, c_2), \lambda\overline{x}.\hat{m}_2(\lambda x_2.\hat{m}_1(\lambda x_1.\overline{x}(M^r(x_1, x_2)))) \rangle$

This equivalence shows that the f rule gives priority to the first subexpression, e_1, and g gives priority to the second subexpression, e_2. The meaning operation M_f^r corresponds to the second schema in (9) as well as (18a), and M_g^r corresponds to (18b).

For rules of arity $n = 3$, $n! = 6$, and so on.

Clearly G and $\overline{G}$ are strongly equivalent syntactically, since their syntactic components are identical. They are also strongly equivalent semantically in the following sense: Let e be an expression of category c associated with a meaning m that is licensed by rule r in G. Then for every corresponding $\hat{m}$ that $\overline{G}$ assigns as a possible meaning to e, $\hat{m}(\lambda x.x) = m$. (This theorem is analogous to Plotkin's (1975) Simulation Theorem proving that a particular CPS transform captures the meaning of the original program.) In order to prove this claim, I will prove a slightly more general proposition.

Lemma: For any function g, $\hat{m}(\lambda x.g(x)) = g(m)$.

Proof: The proof involves recursion on the syntactic structure of e. The base case is when r is arity 0, i.e., e is a lexical item. Instantiating the continuation schema, the meaning $\overline{G}$ associates e with is $\hat{m} = \lambda\overline{x}.\overline{x}(m)$. Therefore $\hat{m}(\lambda x.g(x)) = [\lambda\overline{x}.\overline{x}(m)](\lambda x.g(x)) = g(m)$, and the claim is true. Next, consider the case in which r is of arity 1. Assume that G associates e_1 with the meaning m_1 and that $\overline{G}$ associates e_1 with the meaning $\hat{m}_1$. Then G associates e with the meaning $m = M^r(m_1)$. There is exactly one possible choice for f (namely, the identity function, so that $f(1) = 1$), and we have

$$\hat{m} = \lambda\overline{x}.\hat{m}_{f(1)}(\lambda x_{f(1)}.\overline{x}(M^r(x_1))) = \lambda\overline{x}.\hat{m}_1(\lambda x_1.\overline{x}(M^r(x_1)))$$

Let g be an arbitrary function. Then

$$\begin{aligned}\hat{m}(\lambda x.g(x)) &= [\lambda\overline{x}.\hat{m}_1(\lambda x_1.\overline{x}(M^r(x_1)))](\lambda x.g(x))\\ &= \hat{m}_1(\lambda x_1.[\lambda x.g(x)](M^r(x_1)))\\ &= \hat{m}_1(\lambda x_1.g(M^r(x_1)))\end{aligned}$$

Choose $g' = \lambda x.g(M^r(x))$. Then $\hat{m}_1(\lambda x_1.g(M^r(x_1))) = \hat{m}_1(\lambda x_1.g'(x_1))$. By the recursive assumption,

$$\begin{aligned}\hat{m}_1(\lambda x_1.g'(x_1)) &= g'(m_1) = [\lambda x.g(M^r(x))](m_1)\\ &= g(M^r(m_1)) = g(m).\end{aligned}$$

Thus the claim holds for expressions licensed by rules of arity 1. If e is licensed by a rule r of arity 2, e has exactly two immediate subconstituents,

e_1 and e_2. Assume that G associates e_1 with meaning m_1 and e_2 with meaning m_2, and that $\overline{G}$ associates e_1 with meaning $\hat{m}_1$ and e_2 with meaning $\hat{m}_2$. Then G associates e with the meaning $m = M^r(m_1, m_2)$. There are two possible choices for f. First consider when f is the identity function, so that $f(1) = 1$ and $f(2) = 2$. Instantiating the continuation schema, we have

$$\begin{aligned}\hat{m} &= \lambda\overline{x}.\hat{m}_{f(1)}(\lambda x_{f(1)}.\hat{m}_{f(2)}(\lambda x_{f(2)}.\overline{x}(M^r(x_1, x_2))))\\ &= \lambda\overline{x}.\hat{m}_1(\lambda x_1.\hat{m}_2(\lambda x_2.\overline{x}(M^r(x_1, x_2))))\end{aligned}$$

Once again, let g be an arbitrary function. Then

$$\begin{aligned}\hat{m}(\lambda x.g(x)) &= [\lambda\overline{x}.\hat{m}_1(\lambda x_1.\hat{m}_2(\lambda x_2.\overline{x}(M^r(x_1, x_2))))](\lambda x.g(x))\\ &= \hat{m}_1(\lambda x_1.\hat{m}_2(\lambda x_2.g(M^r(x_1, x_2))))\end{aligned}$$

Choose $g' = \lambda x.g(M^r(x_1, x))$. By the recursive assumption applied to $\hat{m}_2$,

$$\begin{aligned}\hat{m}(\lambda x.g(x)) &= \hat{m}_1(\lambda x_1.\hat{m}_2(\lambda x_2.g'(x_2)))\\ &= \hat{m}_1(\lambda x_1.g'(m_2)) = \hat{m}_1(\lambda x_1.g(M^r(x_1, m_2)))\end{aligned}$$

Now choose $g'' = \lambda x.g(M^r(x, m_2))$. By the recursive assumption applied to $\hat{m}_1$, we have

$$\hat{m}(\lambda x.g(x)) = \hat{m}_1(\lambda x_1.g''(x_1)) = g''(m_1) = g(M^r(m_1, m_2)) = g(m)$$

Analogous reasoning holds for the other choice of f, on which $f(1)= 2$ and $f(2) = 1$. In general, for any number of arguments n and for any choice of permutation function f, we can choose functions g, g', . . . in a way that allows us to apply the recursive assumption from the innermost continuation out, i.e., in the order $\hat{m}_{f(n)}$, $\hat{m}_{f(n-1)}$, . . . , $\hat{m}_{f(1)}$. □

Theorem (Simulation): $\hat{m}(\lambda x.x) = m$.

Proof: Choose $g = (\lambda x.x)$ and the result follows immediately from the lemma. □

Of course the semantic equivalence between G and $\overline{G}$ does not hold in general if we add lexical items or composition rules to $\overline{G}$ that are not related to the basic grammar by the continuation schema. Indeed, adding rules like those in (13) is what allows quantificational readings and scope ambiguity.

6. Compositionality

As discussed above, a continuized grammar is compositional in the sense that the meaning of a complex syntactic constituent is a function only of the meanings of its immediate subconstituents and the manner in which they are combined.

Is compositionality an empirical issue, or formally vacuous and therefore merely a methodological preference? A number of mathematical arguments purport to prove under one set of assumptions or another that any syntax can be associated with literally any set of meanings in a compositional fashion. In particular, Janssen (1986) proves that an arbitrary meaning relation can be embodied by a compositional grammar if we are allowed arbitrarily abstract syntactic analyses (see comments of Westerståhl 1998). In other words, allowing LF representations to differ in unconstrained ways from surface syntax removes all empirical force from assuming compositionality. This is the sense in which LF-based theories of quantification such as QR weaken compositionality. Certainly anyone with a strong commitment to compositionality will prefer a theory on which deviations from surface syntax are kept to an absolute minimum. The ideal is what Jacobson (e.g., 1999) calls Direct Compositionality, in which each surface syntactic constituent has a well-formed denotation and there is no appeal to a level of Logical Form distinct from surface structure. We shall see that continuations are compatible with Direct Compositionality.

Zadrozny (1994) proves a complementary theorem: holding the syntax constant, if the denotations are allowed be abstract in certain ways, once again it is possible to provide a given syntax with any desired meaning relation in an allegedly compositional fashion. More specifically, Zadrozny's construction replaces normal meanings with functional meanings. Should we worry, then, since continuizing a grammar replaces normal meanings with functions? No. Dever (1999) shows that Zadrozny's result crucially depends on the fact that Zadrozny's denotations take the expression itself as an argument. Since in Zadrozny's construction the meaning of an expression depends in part on its form (rather than on just the denotations of its subconstituents), it is no wonder that it has more than compositional power.

Pelletier (1994) explains one way that compositionality can clearly be falsified, which establishes that compositionality is empirically substantive. If there were a natural language in which ϕ and ψ were two distinct expressions that meant the same thing, and $F(\phi)$ and $F(\psi)$ were well-formed syntactic expressions that meant different things (i.e., $[\![F(\phi)]\!] \neq [\![F(\psi)]\!]$), there simply would be no compositional way of assigning meanings to these expressions. For instance, to repeat a standard example, if you believe that the sentences *Hesperus is a planet* and *Phosphorus is a planet* denote the same proposition, and that there is a unique series of syntactic operations that allows construction of both of the sentences *Jeremy believes that Hesperus is a planet* and *Jeremy believes that Phosphorus is a planet*, and that these two more complex sentences are capable of having different truth values – then you believe that English does not have a compositional semantics.

In recognition of this reasoning, Dever (1999, p. 314) imposes a requirement that a theory is compositional only if the meaning of complex expressions remains constant under substitution of subexpressions with equivalent meanings. Zadrozny's allegedly compositional construction fails to meet this requirement, so by Dever's criterion, Zadrozny's meaning relation fails to be compositional. In any case, the continuized grammars considered in this paper clearly satisfy Dever's substitutability requirement.

However, there is an aspect of the continuation approach that threatens the spirit of compositionality in a more serious way. Compositionality, at least as Montague formulated it, requires that a syntactic analysis fully disambiguates the expression in question. As we have seen, the denotation of an expression containing quantifiers depends in part on the choice of a permutation function, since the permutation function determines the scope priority of the constituents. Therefore, in order to obey the letter of the law of compositionality, it is necessary for the syntactic analysis to specify the choice of a permutation function. This is easy enough to do on a mechanical level by annotating syntactic constructions with the index of a permutation function. But doing so implicitly claims that quantifier scope ambiguity is a syntactic ambiguity, and the result is a (mild) form of hypothesizing an LF distinct from surface syntax.

The alternative is to admit, contra Montague, that there is such a thing as semantic ambiguity. A number of other researchers have come to such a conclusion. For instance, Dalrymple et al. (1997, p. 229) allow "ambiguities of semantic interpretation"; this point is elaborated by Crouch (1999, p. 48). Pelletier (1994, p. 21) characterizes the most extreme form of this position as follows: "It seems to me that all of these [LF-based treatments of quantification] are rather desperate measures in that they try to invent a syntactic ambiguity when we know perfectly well that in reality there is no syntactic ambiguity. Admittedly there is a *semantic* ambiguity . . ." (emphasis as in original).

What I am suggesting is that a single syntactic formation operation can be associated with more than one semantic interpretation. The resulting notion of compositionality is as follows:

(36) The meaning of a syntactically complex expression is a function only of the meaning of that expression's immediate subexpressions, the syntactic way in which they are combined, **and their semantic mode of composition**.

In the system here, each permutation function determines one "mode of composition." This places the burden of scope ambiguity on something that is neither syntactic nor properly semantic, but at the interface: scope ambiguity is metacompositional.

Modifying the principle of compositionality as in (36) does interfere slightly with Montague's conception of the meaning relation as a homomorphism from a syntactic algebra into a meaning algebra. If maintaining the conception of meaning as a homomorphism seems important, then we need to consider as our disambiguated objects syntactic analyses in combination with a mode of composition. Call these elaborated analyses CONSTRUALS. Then the syntactic algebra induces an algebra on the set of construals in a natural way, and we can use the algebra over the set of construals as the domain of the meaning homomorphism. Regardless of whether it seems advisable to maintain the meaning relation as a homomorphism, the conception of compositionality in (36) does not interfere in the slightest with the degree to which compositionality imposes substantive empirical constraints on meaning relations.

7. Conclusions

The continuation approach to quantification is most similar in methodological outlook to Hendriks' Flexible Types system. Both approaches respect syntactic structure, i.e., they interpret overt syntactic structure directly without recourse to invisible manipulations at LF; neither uses any kind of storage mechanism; and both are strictly compositional.

In the Flexible Types system, the value-raising schema and the argument-raising schema in effect allow a predicate to climb up the type hierarchy as high as necessary in order to swallow as much of its computational future as its arguments need to take scope over. Since scope can be displaced arbitrarily far, the result for Flexible Types is that even simple lexical transitive verbs must be infinitely polysemous.

Perhaps continuization is the underlying generalization that the various type-shifting rules of the Flexible Types system conspire to simulate.[4]

[4] Actually, it appears that the Flexible Types system can simulate higher-order continuations, which means that it is significantly more powerful than the first-order continuations used here.

Alternatively, it is possible to view the type-shifting schemata of the Flexible Types system as a decomposition of continuations into more basic type-shifting operations. One problem with this view is that it is difficult to see how either Argument Raising or Value Raising is simpler than the continuation schema given in section 5.

I have claimed that continuations do not rely on type-shifting. Yet there is an unmistakable flavor of type-shifting throughout the continuation enterprise. One way to say it is that instead of type-shifting expressions, we are type-shifting composition rules. Perhaps an even better way to view it would be to say that it is the entire grammar as a whole that has been shifted. I will make two points in response to this thought. First, once a grammar has been shifted to its continuized version, there is no need to use the unshifted grammar; in contrast, the point of most type-shifting analyses (e.g., Partee and Rooth 1983; Partee 1987) is that both the shifted and the unshifted version of a meaning are needed in different situations.

Second, the proposal here is by no means the only example of type-shifting a grammar as a whole. In the dynamic grammars of Irene Heim, Hans Kamp, and of Jeroen Groenendijk and Martin Stokhof, among others, expression denotations and composition rules are all shifted so that sentences denote update functions on contexts. Thus the various kinds of dynamic semantics are well-known and respectable cases of type-shifting the grammar as a whole.

In fact, dynamic interpretation constitutes a partial continuization in which only the category S has been continuized (rather than continuizing uniformly throughout the grammar, as is done here). In particular, Chierchia (1995, p. 81) presents his dynamic logic in terms very much like what we are calling continuations (though he uses the word ‘continuation’ in an unformalized sense that seems to be slightly different from the one here). In Chierchia’s framework, sentence denotations are expressed by logical forms that contain a placeholder standing for the content of subsequent discourse. As he puts it, “metaphorically speaking, we add to [the interpretation of S] a hook onto which incoming information can be hung.” Obviously, this sounds very much like a continuized meaning.

Thus continuization not only gives rise to the generalized quantifier conception of NP meaning as a special case, it also naturally gives rise to the conception of S meaning as a dynamic context update function. More specifically, since the semantic type of a continuized clause is a function from sentence continuations to truth values (type $\langle\langle t, t\rangle, t\rangle$), this is exactly the sort of meaning required in order to compose sequences of sentences in a discourse in the way that Chierchia advocates.

To summarize: From an empirical point of view, the continuation analysis automatically makes good predictions concerning linear versus inverse scope within NP as well as concerning NP as a scope island, where other accounts have considerable difficulty.

From a methodological point of view, taking the principle of compositionality seriously means preferring analyses in which logical form stays as close to surface syntax as possible. Hendriks’ Flexible Types system shows that there are compositional alternatives on which interpretation proceeds directly from surface syntax, as long as we are willing to introduce radical type-shifting. The continuation analysis developed here provides a second direct-compositional analysis (‘direct’ in Jacobson’s sense) for quantification – moreover, one that does not depend on type-shifting.

Finally, from an explanatory point of view, continuations provide a new and satisfying way of unifying several aspects of nominal quantification: merely stating the truth conditions of quantificational expressions in terms of continuations accounts for the duality of NP meaning, scope displacement, and scope ambiguity. In addition, continuations provide a semantic analysis of generalized coordination that is significantly simpler than other accounts.

8. Appendix: Fragment with Continuations

First I give a direct grammar without quantification that will serve as the input to the Continuation Schema (and will afterward be discarded):

(37)	Syntax	Direct Semantics
	VP → Vt NP	⟦Vt⟧(⟦NP⟧)
	VP → Vs S	⟦Vs⟧(⟦S⟧)
	NP → Det N	⟦Det⟧(⟦N⟧)
	N → N PP	⟦PP⟧(⟦N⟧)
	N → Nr PPof	⟦Nr⟧(⟦PP⟧)
	PP → P NP	⟦P⟧(⟦NP⟧)
	PPof → of NP	⟦NP⟧
	NP → John	**j**
	P → from	**from**
	VP → left	**left**
	Vt → saw	**saw**
	Vs → thought	**thought**
	Nr → friend	**friend-of**
	Det → the	**the**
	N → dog	**dog**

This grammar gives rise to the following continuized grammar by application of the Continuation Schema discussed in section 5. We will need three additional variable symbols: T of type ⟨t, ⟨e, t⟩⟩ for sentential-complement-taking verbs, D of type ⟨⟨e, t⟩, e⟩ for determiners, and M of type ⟨⟨e, t⟩, ⟨e, t⟩⟩ for prepositional phrase nominal modifiers.

(38)

Syntax	Continuized Semantics
VP → Vt NP	$\lambda\overline{P}.\llbracket\text{Vt}\rrbracket(\lambda R.\llbracket\text{NP}\rrbracket(\lambda x.\overline{P}(Rx)))$
VP → Vt NP	$\lambda\overline{P}.\llbracket\text{NP}\rrbracket(\lambda x.\llbracket\text{Vt}\rrbracket(\lambda R.\overline{P}(Rx)))$
VP → Vs S	$\lambda\overline{P}.\llbracket\text{Vs}\rrbracket(\lambda T.\llbracket\text{S}\rrbracket(\lambda p.\overline{P}(Tp)))$
VP → Vs S	$\lambda\overline{P}.\llbracket\text{S}\rrbracket(\lambda p.\llbracket\text{Vs}\rrbracket(\lambda T.\overline{P}(Tp)))$
NP → Det N	$\lambda\overline{x}.\llbracket\text{Det}\rrbracket(\lambda D.\llbracket\text{N}\rrbracket(\lambda P.\overline{x}(DP)))$
NP → Det N	$\lambda\overline{x}.\llbracket\text{N}\rrbracket(\lambda P.\llbracket\text{Det}\rrbracket(\lambda D.\overline{x}(DP)))$
N → N PP	$\lambda\overline{P}.\llbracket\text{N}\rrbracket(\lambda P.\llbracket\text{PP}\rrbracket(\lambda M.\overline{P}(MP)))$
N → N PP	$\lambda\overline{P}.\llbracket\text{PP}\rrbracket(\lambda M.\llbracket\text{N}\rrbracket(\lambda P.\overline{P}(MP)))$
N → Nr PPof	$\lambda\overline{P}.\llbracket\text{Nr}\rrbracket(\lambda R.\llbracket\text{PPof}\rrbracket(\lambda x.\overline{P}(Rx)))$
N → Nr PPof	$\lambda\overline{P}.\llbracket\text{PPof}\rrbracket(\lambda x.\llbracket\text{Nr}\rrbracket(\lambda R.\overline{P}(Rx)))$
PP → P NP	$\lambda\overline{M}.\llbracket\text{P}\rrbracket(\lambda R.\llbracket\text{NP}\rrbracket(\lambda x.\overline{M}(Rx)))$
PP → P NP	$\lambda\overline{M}.\llbracket\text{NP}\rrbracket(\lambda x.\llbracket\text{P}\rrbracket(\lambda R.\overline{M}(Rx)))$
PPof → of NP	$\lambda\overline{x}.\llbracket\text{NP}\rrbracket(\lambda x.\overline{x}(x))$
NP → John	$\lambda\overline{x}.\overline{x}(\mathbf{j})$
P → from	$\lambda\overline{R}.\overline{R}(\mathbf{from})$
VP → left	$\lambda\overline{P}.\overline{P}(\mathbf{left})$
Vt → saw	$\lambda\overline{R}.\overline{R}(\mathbf{saw})$
Vs → thought	$\lambda\overline{T}.\overline{T}(\mathbf{thought})$
Nr → friend	$\lambda\overline{R}.\overline{R}(\mathbf{friend\text{-}of})$
Det → the	$\lambda\overline{D}.\overline{D}(\mathbf{the})$
N → dog	$\lambda\overline{P}.\overline{P}(\mathbf{dog})$

Determiners and choice functions. Note that the basic grammar in (37) analyzes the determiner *the* as a CHOICE FUNCTION (type ⟨⟨e, t⟩, e⟩): a function that maps a nominal property to an individual. For each property P, the choice function denoted by *the* picks out one element of P; for instance, **the**(**dog**) denotes some particular element from the set of dogs. Choice functions make appealing denotations for at least some determiners (Egli and von Heusinger 1995; Kratzer 1998, inter alia).

But what type will a quantificational determiner be? Since all members of a syntactic category have the same semantic type, it will be the same as the type of a continuized direct determiner, i.e., type ⟨⟨⟨⟨e, t⟩, e⟩, t⟩, t⟩.

For instance, given a variable f over choice functions of type ⟨⟨e, t⟩, e⟩, here is a lexical entry of the appropriate type for *every*:

(39) Det→ every $\lambda\overline{D}\forall f.\overline{D}(f)$

This denotation quantifies over choice functions, rather than over individuals. (It is necessary to restrict the quantification to consider only those functions that map each [non-empty] set to a member of that set; these and similar technical issues are discussed in some depth in the choice function literature cited above.) For instance,

$$\llbracket \textit{John saw every man}\rrbracket(\lambda p.p) = \forall f.\mathbf{saw}(f(\mathbf{man}))\mathbf{j},$$

which can be paraphrased as 'For every way of choosing a man, that man was seen by John'.

Things get slightly more complicated for proportional quantificational determiners such as *most*. Instead of denoting a set of sets of individuals as in the (extensional) generalized quantifier approach (e.g., Barwise and Cooper 1981), *most* (and quantificational determiners in general) will denote sets of sets of choice functions. Let C be a variable over sets of choice functions:

(40) a. Det → most $\lambda\overline{D}\exists C \in \mathbf{MOST}\text{: } \forall f \in C.\overline{D}(f)$
b. $\mathbf{MOST} = \{C\text{: } \forall P.|P| < (2 * |\{x\text{: } \exists f \in C.x = f(P)\}|)\}$

An alternative (not pursued here) would be to allow choice functions to pick out mereological sums of individuals.

To complete the fragment, we must add the rules that make S a scope island, as discussed above in section 2.4. In addition, we must provide rules for the lexical NPs and the quantificational determiners, and instantiate the syntactic schema for generalized conjunction for the categories S, VP, Vt, and NP. The following rules, then, constitute the properly quantificational component of the fragment:

(41)

S → NP VP	$\lambda\overline{p}.\overline{p}(\llbracket\text{NP}\rrbracket(\lambda x.\llbracket\text{VP}\rrbracket(\lambda P.Px)))$
S → NP VP	$\lambda\overline{p}.\overline{p}(\llbracket\text{VP}\rrbracket(\lambda P.\llbracket\text{NP}\rrbracket(\lambda x.Px)))$
NP → everyone	$\lambda\overline{x}\forall x.\overline{x}(x)$
NP → someone	$\lambda\overline{x}\exists x.\overline{x}(x)$
Det → every	$\lambda\overline{D}\forall f.\overline{D}(f)$
Det → a	$\lambda\overline{D}\exists f.\overline{D}(f)$
Det → no	$\lambda\overline{D}\neg\exists f.\overline{D}(f)$
Det → most	$\lambda\overline{D}\exists C \in \mathbf{MOST}.\forall f \in C.\overline{D}(f)$
S → S_1 and S_2	$\lambda\overline{p}.\mathbf{and}(\llbracket\text{S}_1\rrbracket\,\overline{p})(\llbracket\text{S}_2\rrbracket\,\overline{p})$
VP → VP_1 and VP_2	$\lambda\overline{P}.\mathbf{and}(\llbracket\text{VP}_1\rrbracket\,\overline{P})(\llbracket\text{VP}_2\rrbracket\,\overline{P})$
Vt → Vt_1 and Vt_2	$\lambda\overline{R}.\mathbf{and}(\llbracket\text{Vt}_1\rrbracket\,\overline{R})(\llbracket\text{Vt}_2\rrbracket\,\overline{R})$
NP → NP_1 and NP_2	$\lambda\overline{x}.\mathbf{and}(\llbracket\text{NP}_1\rrbracket\,\overline{x})(\llbracket\text{NP}_2\rrbracket\,\overline{x})$

The result of combining the rules in (38) and (41) is a context-free grammar with rule-to-rule interpretation that treats quantification, scope displacement, scope ambiguity, and generalized conjunction without movement, storage, type-shifting, or type polymorphism.

For example, here is an analysis for a sentence similar to the sentences in (19), which were designed to show that the continuized grammar automatically allows unbounded scope displacement:

(42) a. Someone saw the friend of the friend of everyone.
b. $\exists x \forall y.\mathbf{saw}(\mathbf{the}(\mathbf{friend\text{-}of}(\mathbf{the}(\mathbf{friend\text{-}of}\ y))))\ x$
c. $\forall y \exists x.\mathbf{saw}(\mathbf{the}(\mathbf{friend\text{-}of}(\mathbf{the}(\mathbf{friend\text{-}of}\ y))))\ x$

Clearly we can insert an arbitrary number of repetitions of the string *the friend of* before *everyone*, and the grammar will produce both a linear scope reading as in (42b), on which there is a single person who saw, for each person y, the unique friend of the unique friend of y; and also an inverse scope reading as in (42c), on which for every person y there is a potentially different person x who saw the friend of the friend of y.

Finally, here are the four logically distinct interpretations provided by the fragment for the sentence *Someone saw a friend of everyone*:

(43) a. $\exists y \exists f \forall x.\mathbf{saw}(f(\mathbf{friend\text{-}of}\ x))\ y$
b. $\exists y \forall x \exists f.\mathbf{saw}(f(\mathbf{friend\text{-}of}\ x))\ y$
c. $\exists f \forall x \exists y.\mathbf{saw}(f(\mathbf{friend\text{-}of}\ x))\ y$
d. $\forall x \exists f \exists y.\mathbf{saw}(f(\mathbf{friend\text{-}of}\ x))\ y$

The variables y, f, and x correspond to the quantificational elements *someone*, *a*, and *everyone*, respectively. As discussed with respect to example (23) in section 3, because the fragment respects the Integrity constraint, it correctly predicts that there should be four scopings instead of the factorial six.

References

Barker, C.: 2001, ‘Integrity: A Syntactic Constraint on Quantificational Scoping’, in K. Megerdoomian and L. A. Bar-el (eds.), *Proceedings of WCCFL 20*, pp. 101–114. Cascadilla Press, Somerville, Mass.

Barwise, J. and R. Cooper: 1981, ‘Generalized Quantifiers in Natural Language’, *Linguistics and Philosophy* **4**, 159–200.

Chierchia, G.: 1995, *Dynamics of Meaning*, the University of Chicago Press, Chicago.

Chierchia, G. and S. McConnell-Ginet: 1990, *Introduction to Formal Semantics*, MIT Press, Cambridge, Mass.

Crouch, R.: 1999, ‘Ellipsis and Glue Languages’, in S. Lappin and E. Benmamoun (eds.), *Fragments: Studies in Ellipsis and Gapping*, pp. 32–67. Oxford University Press, Oxford.

Dalrymple, M., J. Lamping, F. Pereira, and V. Saraswat: 1997, ‘Quantifiers, Anaphora, and Intensionality’, *Journal of Logic, Language, and Information* **6**, 219–273.

Danvy, O. and C. A. Talcott: 1998, ‘Introduction (to a special issue on continuations)’, *Higher-Order and Symbolic Computation* **11**(2), 115–116.

Dever, J.: 1999, ‘Compositionality as Methodology’, *Linguistics and Philosophy* **22**, 311–326.

Egli, U. and K. Heusinger (eds.): 1995, *Choice Functions in Natural Language Semantics*, Arbeitspapier Nr. 71, Fachgruppe Sprachwissenschaft, Universität Konstanz.

de Groote, P.: 2001, ‘Continuations, Type Raising, and Classical Logic’, in R. van Rooy and M. Stokhof (eds.), *Proceedings of the Thirteenth Amsterdam Colloquium*, pp. 97–101. Institute for Logic, Language and Computation, Universiteit van Amsterdam.

Heim, I. and A. Kratzer: 1998, *Semantics in Generative Grammar*, Blackwell, Oxford.

Hendriks, H.: 1988, ‘Type Change in Semantics: The Scope of Quantification and Coordination’, in E. Klein and J. van Benthem (eds.), *Categories, Polymorphism and Unification*, pp. 96–119. ITLI, Amsterdam.

Hendriks, H.: 1993, *Studied Flexibility*, ILLC Dissertation Series, Amsterdam.

Jacobson, P.: 1999, ‘Towards a Variable Free Semantics’, *Linguistics and Philosophy* **22**, 117–184.

Janssen, T.: 1986, *Foundations and Applications of Montague Grammar, Part I: Philosophy, Framework, Computer Science* (CWI Tract 19), Center of Mathematics and Computer Science, Amsterdam.

Keenan, E.: 1987, ‘Semantic Case Theory’, in J. Groenendijk, M. Stokhof, and P. Veltmann (eds.), *Proceedings of the 6th Amsterdam Colloquium*, pp. 109–132. Institute for Logic, Language and Computation, Universiteit van Amsterdam.

Kelsey, R., W. Clinger, and J. Rees (eds.): 1998, ‘The Revised^5 Report on the Algorithmic Language Scheme’, *Higher-Order and Symbolic Computation* **11**, 7–105.

Kratzer, A.: 1998, ‘Scope or Pseudoscope? Are There Wide-Scope Indefinites?’, in S. Rothstein (ed.), *Events and Grammar*, pp. 163–196. Kluwer, Dordrecht.

May, R.: 1985, *Logical Form: Its Structure and Derivation*, MIT Press, Cambridge, Mass.

Meyer, A. R. and M. Wand: 1985, ‘Continuation Semantics in Typed Lambda-Calculi (summary)’, in R. Parikh (ed.), *Logics of Programs – Proceedings*, pp. 219–224. Springer-Verlag, New York.

Montague, R.: 1973, ‘The Proper Treatment of Quantification in English’, in J. Hintikka, J. Moravcsik, and P. Suppes (eds.), *Approaches to Natural Language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics*, pp. 221–242. Reidel, Dordrecht. Also in R. Thomason (ed.), *Formal Philosophy: Selected Papers of Richard Montague*, pp. 247–270. Yale University Press, New Haven, Conn., 1974.

Partee, B. H.: 1987, ‘Noun Phrase Interpretation and Type-Shifting Principles’, in J. Groenendijk, D. de Jongh, and M. Stokhof (eds.), *Studies in Discourse Representation Theory and the Theory of Generalized Quantifiers*, pp. 115–143. Foris, Dordrecht.

Partee, B. H. and M. Rooth: 1983, ‘Generalized Conjunction and Type Ambiguity’, in R. Bäuerle, C. Schwarze, and A. von Stechow (eds.), *Meaning, Use, and Interpretation of Language*, pp. 361–383. Walter de Gruyter, Berlin.

Pelletier, F. J.: 1994, ‘The Principle of Semantic Compositionality’, *Topoi* **13**, 11–24.

Plotkin, G. D.: 1975, ‘Call-by-Name, Call-by-Value and the Lambda-Calculus’, *Theoretical Computer Science* **1**, 125–159.

Reinhart, T.: 1979, ‘Syntactic Domains for Semantic Rules’, in F. Guenthner and S. J. Schmidt (eds.), *Formal Semantics and Pragmatics for Natural Language*, pp. 107–130. Reidel, Dordrecht.

Reynolds, J. C.: 1993, ‘The Discoveries of Continuations’, *Lisp and Symbolic Computation* **6**, 233–247.

Shan, C-C.: 2002, ‘A Continuation Semantics of Interrogatives that Accounts for Baker’s

Ambiguity', in B. Jackson (ed.), *Proceedings of SALT XII*, Cornell University Press, Ithaca, N.Y.

Steedman, M.: 2000, *The Syntactic Process*, MIT Press, Cambridge, Mass.

Westerståhl, D.: 1998, 'On Mathematical Proofs of the Vacuity of Compositionality', *Linguistics and Philosophy* **21**, 635–643.

Zadrozny, W.: 1994, 'From Compositional to Systematic Semantics', *Linguistics and Philosophy* **17**, 329–342.

0108 Department of Linguistics
University of California, San Diego
9500 Gilman Drive
La Jolla, CA 92093
USA
E-mail: barker@ucsd.edu

EXPLAINING CROSSOVER AND SUPERIORITY AS LEFT-TO-RIGHT EVALUATION

CHUNG-CHIEH SHAN AND CHRIS BARKER

Abstract. We present a general theory of scope and binding in which both crossover and superiority violations are ruled out by one key assumption: that natural language expressions are normally evaluated (processed) from left to right. Our theory is an extension of Shan's (2002) account of multiple-*wh* questions, combining continuations (Barker, 2002a) and dynamic type-shifting. Like other continuation-based analyses, but unlike most other treatments of crossover or superiority, our analysis is directly compositional (in the sense of, e.g., Jacobson 1999). In particular, it does not postulate a level of Logical Form or any other representation distinct from surface syntax. One advantage of using continuations is that they are the standard tool for modeling order-of-evaluation in programming languages. This provides us with a natural and independently-motivated characterization of what it means to evaluate expressions from left to right. We give a combinatory categorial grammar that models the syntax and the semantics of quantifier scope and *wh*-question formation. It allows quantificational binding but not crossover, in-situ *wh* but not superiority violations.

1. Crossover as a Processing Constraint

Typically, a quantifier must precede a pronoun in order to bind it. Thus (1a) can mean that each person loves their own mother, but (1b) cannot mean that each person's mother loves them.

(1) a. Everyone$_i$ loves his$_i$ mother.
b. *His$_i$ mother loves everyone$_i$.

At first glance, a processing explanation seems promising: if humans process sentences from left to right, then arriving at the ungrammatical crossover interpretation in (1b) would require postponing the interpretation of the pronoun until encountering the quantifier that binds it. In general, allowing for crossover interpretations would require the processing mechanism to postpone interpretation of pronouns indefinitely just in case a quantifier occurs later. Since doing so would place an extra load on memory, we expect the bound interpretation for (1b) to be at the very least dispreferred.

One challenge for a processing approach is distinguishing the behavior of quantificational binding from quantifier scope. In at least some configurations, quantifier scope also seems to show a left-to-right processing bias:

(2) a. A student called every professor.
b. Every professor called a student.

Date: April 4, 2004.

Thanks to Daniel Büring, Svetlana Godjevac, Chris Potts, and Stuart Shieber. The first author is supported by National Science Foundation Grant IRI-9712068.

For instance, (2a) is more likely than (2b) to be interpreted to mean that the same student spoke with each professor. If quantifiers that are processed earlier take wider scope, then this preference is again expected.

The problem is that if quantifier scope preferences and crossover arise from the same mechanism, we would expect them to behave similarly. But the left-to-right pattern is at best a preference for quantifier scope, while it is a requirement (in most situations) for quantificational binding. In fact, it is possible for a quantifier to take scope over a pronoun that it cannot bind:

(3) *A student of his$_i$ colleague called every professor$_i$.

The quantificational NP *every professor* can take scope over the subject, as long as the pronoun is taken to refer to some deictic individual (say, the Dean). So right-to-left scope, though awkward, is at least possible. But under most circumstances it is impossible to interpret the pronoun as bound by the quantifier, even when the quantifier takes scope over it.[1]

The puzzle for a processing story is that there is no obvious difference between quantificational binding and scope from a processing point of view: it seems as if any logic that makes a backwards binding relationship more difficult presumably ought to make a backwards scope relationship difficult too. The two theoretical questions, then, are: if quantifier scope and crossover are both connected to processing biases, why is the crossover constraint so much stronger? And: when a quantifier does take scope over a pronoun, what prevents the quantifier from binding the pronoun?

1.1. **Solutions that depend on Quantifier Raising (QR).** The traditional way out of this dilemma is to propose multiple stages for the derivation of a sentence, with binding and scoping taking place at different stages. Reinhart's (1983) approach is an influential example, and Büring (2001) provides a recent analysis that uses the same basic strategy. The idea is to postulate two distinct levels of representation: first a level of surface structure at which binding relationships are established by some syntactic relationship based on c-command, then a level of Logical Form at which quantifiers take their semantic scope. This way, even though a quantifier may raise at LF to take scope over a pronoun, it can still only bind the pronoun if it c-commands the pronoun from its surface position. The claim is that crossover is an essentially syntactic phenomenon: quantificational binding depends on syntactic relations at the surface. In contrast, quantifier scope depends on relations at the level of Logical Form.

Analyses of this sort refer crucially to multiple levels of representation.[2] Moreover, this standard treatment is framed in terms of hierarchical structure (c-command) rather

[1] Postal (1993) (see also the discussion in Potts, 2001) points out that crossover interpretations sometimes become better if *own* or *only* is added: ?*A student of his$_i$ own advisor contacted every professor$_i$*. This amelioration effect is poorly understood, and we will not discuss it further here.

[2] Indeed, the very name 'crossover' comes from Postal's (1971) proposal for a general constraint on movement, roughly: an NP may not move across a coindexed pronoun (p. 62). Even some accounts that attempt to state a crossover-like constraint in terms of single representation (e.g., Pesetsky's (1982) path containment condition for superiority) crucially depend on access to a post-QR structure, and therefore also rely on multiple levels of representation.

than linear order.[3] Since syntactic structure in English is predominantly right-branching, c-command mostly coincides with linear order, so that even on a c-command account, quantifiers will in effect only bind pronouns that follow them in surface structure. In this paper, we will suggest that linear order is the correct generalization after all, at least for English and maybe also Japanese (Kuno and Kim, 1994). We leave a discussion of languages in which c-command does not correlate strongly with linear order for another occasion.[4]

1.2. **A new solution based on continuations.** In contrast to QR-based accounts, we present in this paper a new account of quantifier scope and quantificational binding that does not postulate multiple stages of derivation or multiple levels of representation, and in fact does not even mention c-command. On our account, crossover is a consequence of the following processing default:

(4) Evaluate expressions from left to right.

We formally characterize what it means to evaluate expressions from left to right, adopting a concept developed independently for describing the evaluation order of subexpressions in programming language semantics (see Section 7.2).

As formally implemented below, (4) naturally follows from the commonplace assumption that people process (evaluate) expressions that they hear first before those they hear later. There is abundant evidence (e.g., Tanenhaus et al., 1990 and references there) that people begin to process language immediately, i.e., as soon as they hear the beginning of an utterance, and incrementally, building partial interpretations without waiting for the remainder of the utterance. Furthermore, incremental processing seems to integrate information from all grammatical levels, including morphological, syntactic, semantic, and pragmatic. Of course, there are many ways that a grammatical framework could be consistent with incremental processing, but (4) embodies incrementalism in a natural and direct manner.

But if people process utterances from left to right, how could the quantifier in, say, (2a) be evaluated last in the sentence yet take semantic scope over the subject? It seems paradoxical, yet this is exactly how our system works.

1.3. **An informal sketch of the analysis of crossover.** There are four main steps to understanding our account of crossover. The first step presents our treatment of quantifier scope. Our system uses continuations (Barker 2001; 2002a; de Groote, 2001) with an important innovation: Previous continuation-based accounts of quantification involve first-order continuations only and no dynamic type-shifting. In those analyses, scope ambiguity

[3] Higginbotham (1980) is a notable exception: he relies both on quantifier raising and a re-indexing rule that is sensitive to linear order. Since our binding system does not use indices, we could not restate Higginbotham's re-indexing rule even if we wanted to; nevertheless, we affirm his suggestion that linear order plays an irreducible role in determining quantificational binding relationships.

[4] Reinhart's (1983) classic arguments in favor of hierarchical relations and against linear order fall into three main classes. The first involves non-quantificational anaphora (*Near him, Dan saw a snake*). In this paper, we consider only quantificational binding, which behaves differently from coreference in general. The second involves preposed phrases of various types (*Thinking about his problems, everyone got depressed*, see also Higginbotham (1980)), which Reinhart suggests may be due to "a later stylistic rule" that preposes after binding relation have been established. In this paper, we do not provide an account of stylistic preposing, in which a fronted constituent contains a bound element. (However, we do analyze binding *by* a fronted constituent, and crossover effects there, in Section 6.) The third set of cases involve c-command from one argument of a verb into another (*?You may show his file to each patient who wants to see them*). Reinhart gives each example of this type one question mark, and we are content for now to categorize them as crossover violations.

arises because expressions can be evaluated in a variety of orders—the earlier a quantifier gets evaluated, the wider its scope. In the present theory, which implements higher-order continuations (Danvy and Filinski, 1990) using dynamic type-shifting, expressions are evaluated strictly left-to-right. Scope ambiguity arises because quantifiers can operate at a variety of continuation levels—the higher level a quantifier operates at, the wider its scope.[5]

The second step is understanding how a deeply-embedded expression such as a free pronoun can affect the semantic type of larger expressions in which it is embedded.

(5) a. Mary saw him. ⟨e,t⟩
b. She saw him. ⟨e,⟨e,t⟩⟩

For example, consider the difference between (5a) and (5b). As in other variable-free systems (e.g., Jacobson, 1999), we do not assign meanings to these sentences by relativizing truth conditions to an assignment function. Instead, the single-pronoun sentence (5a) denotes a function from individuals to truth values: the denotation of (5a) is interpreted as a function from a choice for the identity of the referent of the pronoun to the proposition that Mary saw that individual. The double-pronoun sentence (5b), on the other hand, denotes a relation between individuals. The addition of each pronoun increases the arity of the relation denoted by the matrix sentence by one. Thus a deeply-embedded pronoun can control the type of the larger expression in which it is embedded.

The third step is to provide meanings for quantificational expressions that let them bind pronouns. Since a bindable pronoun increases the arity of an expression by one, the binding version of a quantifier decreases it back again.[6]

The fourth and final step is to see how choosing the order of evaluation explains crossover. In a nutshell, it works like this: the denotation of each expression will enter into the composition at a particular continuation level, as determined by lexical meaning in combination with an applications of the type-shifting rules. For two expressions at the same level, the left-to-right evaluation order gives priority to the expression on the left for both scoping and binding: the expression on the left must take scope over the one on the right, and if there is a binding relationship, the expression on the left must be the binder rather than the pronoun. The nature of continuations is such that quantifiers can take scope over expressions to their left by shifting to a higher level of continuations: the higher the level, the wider the scope. But for a quantifier to bind a pronoun, the quantifier and the pronoun must occupy the same level; it naturally falls out of our system that shifting the quantifier to a higher level does not help it bind. Therefore a quantifier cannot bind any pronoun that precedes it.

Thus we will propose a general mechanism for quantifier scope that has a left-to-right bias, that nevertheless allows for inverse quantifier scoping, but that does not allow for inverse quantificational binding (i.e., that does not allow crossover violations).

[5] Hendriks' (1988; 1993) Flexible Types is another dynamic type-shifting solution that is equivalent to higher-order continuations, although the connection to continuations is not made by Hendriks.

[6] Locating binding in the quantifier differs from Jacobson's strategy, since she places the burden of binding on the predicates that take the pronouns and quantifiers as arguments. For instance, in (1a), binding is accomplished by type-shifting *loves*, not by type-shifting the quantifier in the way we propose here. Jacobson's strategy leads to a different analysis of crossover that is hard to phrase in terms of evaluation order (Barker, 2002b).

1.4. **A note on some varieties of crossover.** The literature cross-classifies crossover into four varieties, by whether the binder or the pronoun is or is not embedded within a containing NP. When it is the status of the pronoun that is in question, the varieties are called 'strong crossover' versus 'weak crossover':

(6) a. *He$_i$ likes every man$_i$. strong crossover
b. *His$_i$ mother likes every man$_i$. weak crossover

Examples like (6a), in which the pronoun is unembedded, is usually judged to be significantly worse than their embedded counterparts like (6b). Thus the crossover types are called 'strong' and 'weak' respectively.

Our analysis rules out both strong and weak crossover without distinguishing between them. Thus any genuine difference in strength must arise from some factor independent of what rules out crossover in general. For instance, only in strong crossover does the pronoun c-command the quantificational NP. Thus strong crossover configurations may violate independent binding constraints in addition to whatever rules out crossover violations. In any case, we will not say anything further about the difference between strong crossover and weak crossover.

When it is the unembedded versus embedded status of the binder that is in question, the varieties are called 'primary crossover' versus 'secondary crossover' (Postal, 1993):

(7) a. *He$_i$ likes every man$_i$. primary crossover
b. *He$_i$ likes every man$_i$'s mother. secondary crossover

When the quantificational NP is unembedded, we have primary crossover, and when the quantificational NP is embedded within a larger NP, we have secondary crossover. Some theories of quantification have great difficulty allowing embedded quantifiers to bind pronouns they don't c-command (see Heim and Kratzer, 1998 (p. 234), Büring, 2001 and Barker, 2002b for discussion). In our theory of quantificational binding, secondary binding (what Büring calls "binding out of DP", e.g., *Everyone$_i$'s mother loves him$_i$* or *Some person from every city$_i$ loves it$_i$*) falls out without special stipulation, and secondary crossover (both strong and weak) likewise follows from the same explanation that holds for primary crossover.

One type of crossover that we will not discuss at all is what Lasnik and Stowell (1991) call 'weakest crossover', which they argue involves binders that are not genuinely quantificational. In this paper, we will consider only cases in which the potential binder is either a quantificational NP or else a *wh*-phrase (see immediately below), both of which qualify as genuinely quantificational in Lasnik and Stowell's terms.

Thus this paper proposes a unified explanation of strong and weak primary and secondary crossover. What's more, the same principle (4) that accounts for these varieties of crossover also accounts for superiority.

2. Superiority and Its Relation to Crossover

'Superiority' is the name for the fact observed by Kuno and Robinson (1972) (p. 474) that moving a *wh*-phrase across an in-situ *wh*-phrase results in ungrammaticality.

(8) a. Who ate what?
b. *What did who eat __?

The term "superiority" comes from Chomsky's (1973) proposal for a general constraint that prohibits moving a phrase if a superior (roughly, a c-commanding) phrase could have been moved instead. As a result, (8b) is ruled out because the *wh*-phrase in object position moved when the superior *wh*-phrase in subject position could have. Unlike the traditional crossover case in (1), here the quantificational element (the *wh*-phrase) moves overtly rather than covertly, and the crossed element is an in-situ *wh*-phrase rather than a bound pronoun.

For many years the dominant explanation for superiority attempted to reduce superiority to the ECP, which is a constraint on legitimate LF representations. Unfortunately, as Hornstein (1995) (p. 124) and others point out, the ECP analysis has empirical shortcomings. Because the ECP distinguishes subjects from other argument positions, it can account for the basic contrast in (8), but it fails when both *wh*-phrases correspond to non-subject argument positions.

(9) a. Who did Tom persuade $\_\_$ to buy what?
b. *What did Tom persuade who to buy $\_\_$?

Some more recent accounts of superiority recast it on Minimalist assumptions as a combination of Greed, Procrastinate, Shortest Move, along with a transderivational constraint; others, as a combination of Attract with the Minimal Link Condition; still others, as an Optimality Theoretic interaction among a constraint requiring that Spec of CP be filled by a *wh*-phrase, a constraint that *wh*-phrases raise for interpretation, and a general prohibition against movement. The details of these accounts are fairly intricate, and we will not discuss them further here, except to note that none of them connects superiority with crossover, and none of them refers to linear order.[7]

In contrast, building on Chierchia's (1991; 1993) analysis of pair-list readings, Hornstein (1995), Dayal (1996), and Comorovski (1996) separately argue that superiority reduces to weak crossover.

(10) Who$_i$ $\_\_$ bought [*pro*$_i$ what]?
*What$_i$ did [*pro*$_i$ who] buy $\_\_{}_i$?

For instance, on Hornstein's analysis, an in-situ *wh*-phrase denotes a Skolem function of type $\langle \mathtt{e}, \mathtt{e} \rangle$ whose argument corresponds to a silent pronoun (*pro*). If we stipulate that in-situ *wh*-phrase pronominals must be bound by the raised *wh*-phrase, then superiority violations create a classic weak crossover configuration as shown in (10).

We propose not to reduce superiority to crossover or vice versa, but that crossover and superiority arise through somewhat different binding mechanisms, both of which are sensitive to order of evaluation. That is what accounts for (a significant portion of) their mutual resemblance. Unlike the ECP analysis, our account does not care whether the elements involved are subjects, and naturally generalizes to configurations like (9). Unlike Hornstein's and similar analyses, our account does without covert syntax entirely: there is no distinct level of LF, no covert pronominals, no syntactic indexing, and no distinction between the meaning of a raised *wh*-phrase versus an in-situ *wh*-phrase. In fact, there is no stipulation at all beyond providing *wh*-phrases with a basic syntactic category and truth conditions, since superiority falls out from the independently-motivated theory of scope given in Section 3.[8]

2.1. An informal sketch of the analysis of superiority.

Although we agree with Hornstein, Dayal, and Comorovski that crossover and superiority share an essential explanatory element (in our case, sensitivity to order of evaluation), we do not attempt to reduce superiority to crossover per se. In fact, for us crossover violations and superiority violations arise from different binding mechanisms: Crossover arises from the fact that binders must be evaluated before the pronoun that they bind. Superiority, as we will see, arises from the fact that a raised *wh*-word can only "bind" its *wh*-trace if the *wh*-trace is evaluated before any in-situ *wh*-word.[9] Thus our account of superiority is similar to that for crossover but not identical.[10]

Consider first a single *wh*-question such as *Who$_i$ did Mary say $\_\_{}_i$ bought something?* In order for the *wh*-phrase to "bind" its trace, the complement to the raised *wh*-phrase must provide access to the embedded trace; in semantic terms, the complement must denote a function whose first argument corresponds to the trace. Thus *Mary say $\_\_$ bought something* might denote $\lambda t.\, \mathbf{say}(\exists y.\mathbf{buy} y t)\mathbf{m}$, where the first argument (λt) abstracts over the argument position corresponding to the trace. In scope terms, the trace must receive widest scope.

In particular, if there is also an in-situ *wh*-phrase, the trace must outscope the in-situ *wh*-phrase. Thus in the multiple-*wh*-phrase *Who did Mary say $\_\_$ bought what?*, the constituent *Mary say $\_\_$ bought what* must denote the function $\lambda t.\, \mathbf{what}(\lambda y.\, \mathbf{say}(\mathbf{buy}\ y\ t)\ \mathbf{m})$, in which the λ-abstract corresponding to the trace (λt) has scope over the denotation of the in-situ *wh*-phrase.

Now consider a superiority violation such as **What$_i$ did who buy $\_\_{}_i$?*: an in-situ *wh*-word (*who*) intervenes between the raised *wh*-word (*what*) and its trace. By the reasoning just given, the trace must take scope over *who*. But by the reasoning in Section 1.3, in order for the trace to take scope over something to its left, the trace must type-shift to operate at a higher continuation level. Thus, the trace and the in-situ *wh*-word must operate at two different continuation levels. The problem, which falls out from the combinatorics of continuations, is that two *wh*-words cannot operate at different continuation levels in a complete derivation for a clause. In this, *wh*-words behave differently from (normal) quantificational NPs such as *everyone*. The reason for this difference stems directly from the difference in their meanings: if a quantificational NP is added to (what would otherwise be) a declarative statement, the matrix sentence remains a statement; but if a *wh*-phrase is added to (what is otherwise) a declarative statement, the statement becomes a question. The lowering rule for continuation levels—(16b) below—specifically targets statement meanings (type $\mathtt{t}$) and does not apply to any question type.

The upshot of the above reasoning is that a *wh*-trace must be evaluated before any in-situ *wh*-phrase for the same clause. Given left-to-right evaluation, this means the trace must precede any in-situ *wh*-phrases. In other words, given left-to-right evaluation, an intervening in-situ *wh*-word fatally disrupts the binding relation between a raised *wh*-word and its trace.

3. A General Theory of Scope, Quantificational Binding, and Wh-questions

In this section we present a general system for describing quantifier scope, quantificational binding, and *wh*-question formation. We say that the system is general not because it is comprehensive—indeed, it contains only what is necessary to discuss crossover and

[7] See Dayal, 2003 for a survey of explanations for superiority.

[8] Moreover, because our analysis is based on Shan's (2002), it has all of the virtues of that analysis, including an account of Baker's (1970) ambiguity, which we will not discuss here.

[9] It may seem odd to have a variable-free system that obeys Direct Compositionality (see Jacobson, 1999) yet allows empty categories called (inaccurately) 'traces'. We find that having traces in the examples makes the discussion much easier to understand. In footnote 17 on page 17, we remark on how traces can be eliminated in favor of a type-shifting rule if desired.

[10] O'Neil (1993) also proposes that crossover and superiority be analyzed as special cases of a more general consideration, namely his Generalized Scope Marking Condition. However, his two special cases are individually stipulated and lack independent motivation.

superiority—but because it contains no assumptions, mechanisms, or stipulations specific to crossover or superiority. That is, everything in this section is motivated by the simplest considerations of scope, binding, and *wh*-question formation.

We present below a compositional fragment of English couched in a combinatory categorial grammar. Our analysis has close similarities, as well as important differences, compared with other combinatory categorial grammars (and systems equivalent to combinatory categorial grammars), including analyses due to Hendriks (1993), Jacobson (1999), and Steedman (2000), as well as with certain type-logical accounts discussed by Dowty (1991). Like Hendriks' and Steedman's, our system handles scope; like Jacobson and the accounts discussed by Dowty, it handles quantificational binding; in addition, it also extends naturally to handle single- and multiple-*wh* questions.

Though we are indebted to the work just mentioned, the main ideas below are not developments or elaborations of those accounts. Rather, our approach is more directly related to work in the computer science literature concerning type shifting (Hindley and Seldin, 1986), continuations (Meyer and Wand, 1985), and higher-order control (Danvy and Filinski, 1990)—this is 'control' in the computer science sense of order-of-evaluation. For motivation and intuitions concerning continuations as applied to natural language, see Barker, 2001, 2002a, de Groote, 2001, and Shan, 2002.

It is particularly striking to note the resemblance between our approach and Jacobson's variable-free account of binding, as formulated in, e.g., Jacobson, 1999. For example, one crucial element of her account is that pronouns denote the identity function, and that is true of our account as well. Unlike Jacobson, however, we did not start with a desire to eliminate variables and then build a system that made that desire come true; rather, our starting point was the idea of using continuations to describe scope, which required figuring out how pronouns and binding could fit into a continuation-based interpretation strategy. As we will see below, this naturally leads to a system in which pronouns denote the identity function. The fact that both Jacobson's and our approaches arrive at the same idea of pronouns as denoting the identity function is not a case of theoretical copying, but an instance of homologous convergence—which makes the idea that much more interesting.

The remainder of this section is a detailed and somewhat painstaking explanation of how the grammar works. On a first reading, an impatient reader might prefer to skim this section, glancing over the examples to get a feel for what a derivation looks like, and perhaps considering the summary rule set in Section 3.6 before going on to the discussions in the later sections, returning to Section 3 when more detail is required.

3.1. **Syntactic categories and semantic types.** We use the standard Montagovian extensional type system, in which there are two basic types: $\mathtt{e}$, the class of individuals, and $\mathtt{t}$, the class of truth values. Complex types are written $\langle\alpha,\beta\rangle$, which names the class of functions from objects of type α into the class of objects of type β. For instance, an expression of type $\langle\mathtt{e},\mathtt{t}\rangle$ is a function from individuals to truth values (as usual).

In the spirit of the Curry-Howard isomorphism, and in the traditions of Montague, categorial grammar, and type-logical grammars, we will keep the mapping between syntactic category and semantic type as transparent and direct as possible. We have two basic syntactic category labels, $\mathtt{e}$ and $\mathtt{t}$. Complex syntactic categories are of the form A/B, $A\backslash B$, $A \Rightarrow B$, $A \sim B$, $A \rightarrow B$, $A \rhd B$, or $A \mathbin{?} B$, where A and B are again syntactic categories. No matter what the connective symbol in the middle is, the semantic type of expressions in each of these categories is $\langle\alpha,\beta\rangle$, where α is the semantic type of expressions in category A, and β is the semantic type of expressions in category B. The differences between the various connectives, then, play a role in constraining syntactic combination (compare with Montague's (1974) use of categories of the form A/B versus $A/\!/B$ for distinguishing syntactically between categories with the same semantic type in PTQ).

We will introduce and comment on each of the connectives separately in due course. In addition to making syntactic distinctions, the various connectives also track conceptual differences, distinguishing properties from continuations, or properties from questions, and so on. Even if these categories are modeled by the same semantic objects, it is important to remain aware of their different status, both for the sake of mental hygiene, and also in order to understand their role in the overall grammar.

We provide exactly one rule that allows two expressions to combine to form a syntactically complex expression:

(11) Syntax: $A \Rightarrow B$ followed by A yields B
Semantics: f x fx

This rule says that a phrase of category $A \Rightarrow B$ followed by a phrase of category A can be combined into a complex phrase of category B.[11]

To support the usual forward and backward combination rules, we provide two category-shifting rules, F and B:

(12) a. Forward combination (F): $(A/B) \Rightarrow A \Rightarrow B$
Semantics: $\lambda f x.\, f x$
b. Backward combination (B): $A \Rightarrow (A\backslash B) \Rightarrow B$
Semantics: $\lambda x f.\, f x$

Here and throughout the paper we adopt the convention from computer science discussions that categories associate to the right but expressions associate to the left. Thus '$(A/B) \Rightarrow A \Rightarrow B$' in (12a) above is short for '$(A/B) \Rightarrow (A \Rightarrow B)$' (right association), but 'F saw Mary' and '**saw m j**' in (15a) below stand for '(F saw) Mary' and '(**saw m**) **j**', respectively (left association). The schema in (12b), then, says that any expression of category A with denotation a is also of category $(A\backslash B) \Rightarrow B$ with denotation $(\lambda x f.\, f x)a = \lambda f.\, f a$, for any choice of category B.[12]

The best way to illustrate the way this system works is with a simple example. First we need some lexical entries as follows:[13]

(13)

	Expression	*Category*	*Denotation*	*Description*
a.	Mary	e	**m**	non-quantificational NP
b.	mother	e\e	**mother**	relational noun
c.	left	e\t	**left**	intransitive verb phrase
d.	saw	e/e\t	**saw**	transitive verb
e.	gave	e/e/e\t	**gave**	ditransitive verb
f.	thought	t/e\t	**thought**	verb with sentential object

We can now derive the sentence *Mary left.*

[11] Because the combination rule in (11) only allows complex categories on the left to combine with simpler categories on the right, it might seem that it imposes a left-to-right processing bias. However, as we shall see in Section 7, the form of (11) is irrelevant to our notion of evaluation order, which depends only on the category-shifting rules in (16).

[12] We call lexical entries like F and B "schemata" because their syntactic categories contain variables like A and B that must be resolved to specific categories by unification. These type-shifting rules are exactly the same kind of object as Hendriks' type-shifting rules, Jacobson's combinatory operators, or Steedman's operators. They can also be viewed as inference rules in a type-logical framework.

[13] We sometimes use other lexical items on the assumption that it is obvious which entry above should serve as a pattern. For instance, *John* patterns like *Mary*, and *friend* like *mother*.

(14)
$$\begin{array}{lll} \text{B} & \text{Mary} & \text{left}\\ \mathtt{e}\Rightarrow(\mathtt{e\backslash t})\Rightarrow\mathtt{t} & \mathtt{e} & \mathtt{e\backslash t}\\ \hline \multicolumn{2}{l}{(\mathtt{e\backslash t})\Rightarrow\mathtt{t}} & \\ \hline \multicolumn{3}{l}{\mathtt{t}} \end{array}$$

Or, more succinctly, *(B Mary) left*, which denotes **left m** after beta-reduction.

Following are some more example analyses with their denotations.[14]

(15) a.
$$\begin{array}{lllll} \text{B} & \text{John} & \text{F} & \text{saw} & \text{Mary}\\ \mathtt{e}\Rightarrow(\mathtt{e\backslash t})\Rightarrow\mathtt{t} & \mathtt{e} & (\mathtt{e/e\backslash t})\Rightarrow\mathtt{e}\Rightarrow(\mathtt{e\backslash t}) & \mathtt{e/e\backslash t} & \mathtt{e}\\ \cline{1-2}\cline{3-4} \multicolumn{2}{l}{(\mathtt{e\backslash t})\Rightarrow\mathtt{t}} & \multicolumn{3}{l}{\mathtt{e}\Rightarrow(\mathtt{e\backslash t})}\\ \cline{3-5} & & \multicolumn{3}{l}{\mathtt{e\backslash t}}\\ \hline \multicolumn{5}{l}{\mathtt{t}} \end{array}$$

B John (F saw Mary) = **saw m j**

b. B (B John's mother) left = **left** (**mother j**)

c. B Mary (F thought (B John left)) = **thought** (**left j**) **m**

Thus F and B re-express the usual categorial connectives \ and / in terms of ⇒. We have taken this unusual step because we will need analogs of \ and / that handle higher-order continuation levels. Implementing F and B as category-shifting rules lets them undergo category shifting themselves before participating in a higher-order derivation like (20) below.

Following general practice, we often refer to category-shifting operators like F and B as type-shifting rules. This is correct, but it is important to remember that these rules also shift syntactic categories.

3.2. **Continuations and quantifier scope.** To account for quantifier scope displacement and quantifier scope ambiguity, we need four new category-shifting operators in addition to F and B. These new operators manipulate layers of continuations. These rules are similar (but not identical) to rules defined by Shan (2002).

(16) a. LIFT (L): $A \Rightarrow ((A \sim B) \to B)$
Semantics: $\lambda xc.\, cx$

b. EVAL (E): $((\mathtt{t} \sim \mathtt{t}) \to A) \Rightarrow A$
Semantics: $\lambda X.\, X(\lambda x.\, x)$

c. SKIP (S): $(A \Rightarrow B) \Rightarrow ((A \sim D) \to E) \Rightarrow ((B \sim D) \to E)$
Semantics: $\lambda XYc.\, Y(\lambda y.\, c(Xy))$

d. META (M): $(A \Rightarrow B \Rightarrow C) \Rightarrow$
$((A \sim E) \to F) \Rightarrow ((B \sim D) \to E) \Rightarrow ((C \sim D) \to F)$
Semantics: $\lambda XLRc.\, L(\lambda l.\, R(\lambda r.\, c(Xlr)))$

The connective '∼' labels continuation objects. The category '$A \sim B$' can be read as 'a continuation expecting an argument of category A and returning a result of category B'. Similarly, the connective '→' labels simple functions, so '$A \to B$' is a function from objects corresponding to category A to objects corresponding to category B.

In order to exploit continuations, we must provide lexical items that explicitly mention them.

[14] For simplicity, we treat the possessive clitic *'s* as syntactically and semantically vacuous ($A\backslash A : \lambda x.\, x$ if you prefer).

(17) everyone $(\mathtt{e} \sim \mathtt{t}) \to \mathtt{t}$ $\lambda c.\, \forall x.\, cx$
someone $(\mathtt{e} \sim \mathtt{t}) \to \mathtt{t}$ $\lambda c.\, \exists x.\, cx$

In general, a category of the form $(A \sim D) \to E$ can be thought of as a lifted A. Intuitively, it is an A wrapped in one additional layer of continuation. As a special case, if we let $A = \mathtt{e}$ and $D = E = \mathtt{t}$, then we recover the category $(\mathtt{e} \sim \mathtt{t}) \to \mathtt{t}$, given here as the lexical category of *everyone* and *someone*. Since this is the type of an (extensional) generalized quantifier, Montague's conception of NPs as denoting generalized quantifiers is a special case of continuizing an expression of type e (this point is developed at length in Barker, 2002a).

As will be illustrated shortly, continuation levels are typically established near the beginning of a derivation by LIFT and canceled near the end by EVAL. The two other operators, SKIP and META, allow lifted values to be combined. SKIP matches any unary type-shifting rule and produces a second, more elaborate unary type-shifting rule that operates on lifted values. Similarly, META takes a binary rule and produces a new fancier binary rule that combines lifted values. Both F and B can have META applied to them one or more times, and both L and E can have SKIP applied to them one or more times, producing rules that handle one or more layers of continuation.

To see how the operators in (16) enable higher-order composition, let us try to combine *everyone* (which is intrinsically quantificational) with *left*.

(18) a. everyone
$(\mathtt{e} \sim \mathtt{t}) \to \mathtt{t}$
b. left
$\mathtt{e\backslash t}$

The semantic types here are suitable for functional application, since the verb denotes a function from individuals to truth values, and the NP denotes a function from verb phrase meanings to truth values. However, the syntactic categories cannot be used to instantiate the composition schema in (11), since neither category contains the symbol '⇒'. Nor will application of B or F provide a way to combine the two expressions.

Instead, we must proceed indirectly. The category of *everyone* is that of a lifted e, whereas the category of *left* is unlifted. For *left* to combine with a quantificational NP like *everyone*, it must first be lifted to a suitable type. In general, two expressions can only combine if they are at the same continuation level. To satisfy this requirement in the present case, we need to instantiate the LIFT schema with $A = \mathtt{e\backslash t}$ and $B = \mathtt{t}$:

(19) LIFT(left) = $((\mathtt{e\backslash t}) \sim \mathtt{t}) \to \mathtt{t}$

LIFT generalizes Partee and Rooth's (1983) type-shifting operator of the same name, and plays an analogous role in our analysis of quantification. (It is also homologous to Hendriks' Value Raising and Jacobson's 'lift' operator.) For instance, if *John* has the category e and denotes the individual **j**, then LIFT(John) as a special case has the category $(\mathtt{e} \sim \mathtt{t}) \to \mathtt{t}$ and denotes the generalized quantifier $\lambda c.\, c\,\mathbf{j}$.

The result in (19) of lifting *left* still doesn't match the combination schema in (11), but it is exactly what we will need to apply a first-order combination rule. To arrive at the relevant first-order combination rule, we can apply the META operator to the backward combination rule (12b). The output is

(20) First-order backward combination, M B:
$((A \sim E) \to F) \Rightarrow (((A\backslash B) \sim D) \to E) \Rightarrow ((B \sim D) \to F)$

META B, in other words, is backwards combination lifted to the level of first-order continuations. Second-order backward combination requires M (M B), and so on.

We now instantiate M B with $A = \mathtt{e}$ and $B = D = E = F = \mathtt{t}$:

(21) $((\mathtt{e}\sim\mathtt{t})\to\mathtt{t})\Rightarrow(((\mathtt{e\backslash t})\sim\mathtt{t})\to\mathtt{t})\Rightarrow((\mathtt{t}\sim\mathtt{t})\to\mathtt{t})$

Since the category of *everyone* matches the argument category of (21), we can combine (21) with (18a) using the master combination rule in (11) to get

(22) $(((\mathtt{e\backslash t})\sim\mathtt{t})\to\mathtt{t})\Rightarrow((\mathtt{t}\sim\mathtt{t})\to\mathtt{t})$.

Thus (22) is a valid category for the expression *everyone*. This category does combine with the category of the lifted verb *left* (compare (22) with (19)), to yield the category $(\mathtt{t}\sim\mathtt{t})\to\mathtt{t}$ for the phrase *everyone left*. In other words, we have managed to combine the quantificational subject with a lifted version of the verb phrase using a first-order version of backward combination.

To finish off the derivation, we apply EVAL, which lowers the complex category $(\mathtt{t}\sim\mathtt{t})\to\mathtt{t}$ to $\mathtt{t}$. EVAL is in some sense the inverse of LIFT, and analogous to Partee's (1987) LOWER operation. EVAL is not the exact inverse of LIFT, however, since it specifically requires the category $\mathtt{t}$ instead of a variable over an arbitrary category (though the semantics would be consistent with the more general formulation). This restriction will come into play when we explain superiority in Section 5.

We can summarize the syntactic type-shifting aspect of this analysis using the diagram below, where the categories named by E, M B, and L are elided for brevity.

(23)
$$\begin{array}{llll}
\text{E} & \text{M B} & \text{everyone} & \text{L}\ \ \text{left}\\
 & & (\mathtt{e}\sim\mathtt{t})\to\mathtt{t} & \quad\ \mathtt{e\backslash t}\\ \cline{2-3}\cline{4-4}
 & \multicolumn{2}{l}{(((\mathtt{e\backslash t})\sim\mathtt{t})\to\mathtt{t})\Rightarrow((\mathtt{t}\sim\mathtt{t})\to\mathtt{t})} & ((\mathtt{e\backslash t})\sim\mathtt{t})\to\mathtt{t}\\ \cline{2-4}
 & \multicolumn{3}{l}{(\mathtt{t}\sim\mathtt{t})\to\mathtt{t}}\\ \cline{1-4}
\mathtt{t} & & &
\end{array}$$

More succinctly, using the single-letter abbreviations given in (16), we have the following derivations:

(24) Everyone left. [summary version of (23)]
E ((M B) everyone (L left)) = $\forall x.\,\mathbf{left}\ x$

(25) John saw everyone.
E ((M B) (L John) ((M F) (L saw) everyone)) = $\forall x.\,\mathbf{saw}\ x\ \mathbf{j}$

(26) Everyone's mother left.
E ((M B) ((M B) everyone's (L mother)) (L left)) = $\forall x.\,\mathbf{left}(\mathbf{mother}\ x)$

(27) Everyone's mother's friend left.
E ((M B) ((M B) ((M B) everyone's (L mother's)) (L friend)) (L left))
= $\forall x.\,\mathbf{left}(\mathbf{friend}(\mathbf{mother}\ x))$

These examples show that the grammar as given also derives sentences where a quantificational NP occurs in direct object position or embedded arbitrarily deep within a possessive NP.

Note that the category-shifting rules never affect the order of syntactic combination. Order of combination is determined by the distribution of \ and / connectives in the lexical entries; once, e.g., the lexical entry for a transitive verb like *saw* specifies that it combines first with its direct object and then with its subject, no combinations of the type-shifting rules will enable the verb to combine first with its subject. Put more simply, the type-shifting rules respect the order of syntactic combination established by the lexical entries.[15]

3.3. **Quantifier scope ambiguity.** When a sentence contains more than one quantificational NP, the simplest analysis provides linear scope. In other words, the system has a built-in left-to-right bias, which means that expressions on the left tend to outscope expressions to their right:

(28) Someone saw everyone.
E (M B someone (M F (L saw) everyone)) = $\exists y.\,\forall x.\,\mathbf{saw}\ y\ x$

In order to get inverse scope, it is necessary to lift *everyone* to operate at the level of second-order continuations. The basic idea is that an expression at a given level takes scope over everything at a lower level: the lexical items *everyone* and *someone* come out of the lexicon operating at level 1, and take scope over the rest of a zero-level sentence, so if we lift *everyone* to level 2 but leave *someone* at level 1, then *everyone* will take scope over *someone*, even if *everyone* follows *someone*. Thus lifting to higher levels of continuations overcomes the bias towards linear scope.

So we need to resort to second-order continuations. Before tackling an example with two quantifiers in it, let's work through a simpler example with only one quantifier.

(29) (M (M B)) (L everyone) (L (L left))
$(((\mathtt{t}\sim\mathtt{t})\to\mathtt{t})\sim A)\to A$

Since *everyone* is lexically first-order, *L everyone* is second-order; since *left* is lexically zeroth-order, *L (L left)* is also second-order. The second-order combination rule (i.e., M (M B)) nicely combines the two words, but the result (29) is something more complicated than a truth value, so this derivation is not yet complete.

In previous examples, which involved first-order continuations only, we were able to lower the result category back down to $\mathtt{t}$ by applying EVAL. We can't do that here, because EVAL requires the category $(\mathtt{t}\sim\mathtt{t})\to\mathtt{t}$, whereas the category in (29) is that of a *lifted* $(\mathtt{t}\sim\mathtt{t})\to\mathtt{t}$. Essentially, because EVAL is a one-level rule, it cannot apply directly to a two-level expression; rather, we must first lift EVAL into a two-level rule. This is exactly what SKIP does:

(30) EVAL: $((\mathtt{t}\sim\mathtt{t})\to A)\Rightarrow A$
SKIP: $(A\Rightarrow B)\Rightarrow((A\sim D)\to E)\Rightarrow((B\sim D)\to E)$
SKIP EVAL: $((((\mathtt{t}\sim\mathtt{t})\to A)\sim D)\to E)\Rightarrow((A\sim D)\to E)$

This SKIP EVAL rule—S E for short—applies to (29) exactly, resulting in the category $(\mathtt{t}\sim A)\to A$. Instantiating the category variable A as $\mathtt{t}$, we can then apply plain EVAL, finally yielding a truth value. The complete derivation is:

(31) E ((S E) ((M (M B)) (L everyone) (L (L left)))) = $\forall x.\,\mathbf{left}\ x$

Since there is only one quantifier in this example, there is no opportunity for quantifier scope ambiguity, and so the semantic result is the same as the first-order analysis.

We are now ready to derive inverse scope for the sentence *Someone saw everyone*. As before, we can lift *someone* and *everyone* from first- to second-order by applying LIFT once, and lift *saw* from zeroth- to second-order by applying LIFT twice. These second-order expressions can then be combined using second-order forward and backward combination (i.e., M (M F) and M (M B)):

[15] Respect for syntactic phrasing is by no means a feature of all combinatory categorial grammars, though Hendriks' Flexible Types is a notable exception that does respect syntactic phrasing. In particular, the fragment given in this paper does not include function composition, which is one mechanism that disrupts natural constituency boundaries. Bear in mind, however, that a more complete grammar for a natural language may need to have function composition in order to handle, for example, so-called non-constituent coordination.

(32) M (M B) (L someone) (M (M F) (L (L saw)) (L everyone))
$(((\mathtt{t} \sim \mathtt{t}) \to \mathtt{t}) \sim A) \to A$

To lower this twice-lifted truth value back down to $\mathtt{t}$, we can apply S E followed by E as before.

(33) E (S E (M (M B) (L someone) (M (M F) (L (L saw)) (L everyone))))
$= \exists y.\, \forall x.\, \mathbf{saw}\, x\, y$

Unfortunately, this semantic result is not what we were hoping for: instead of inverse scope, we have derived the same linear scope result we got with the first-order analysis. The reason is that although we have lifted the expressions to the second level, the quantificational force of both *someone* and *everyone* are still operating at the first level. For *everyone* to exploit second-order continuations and outscope *someone*, it must deploy its quantificational meaning at the second level. Lifting *everyone* to *L everyone* merely wraps a first-level quantifier in a passive second-level shell.

The solution is a judicious application of SKIP, this time applied to LIFT instead of EVAL:

(34) LIFT (L): $A \Rightarrow ((A \sim B) \to B)$
SKIP LIFT (S L): $((A \sim D) \to E) \Rightarrow ((((A \sim B) \to B) \sim D) \to E)$

Note that both L and S L turn a first-order expression into a second-order expression: both *L everyone* and *S L everyone* are of category $(((\mathtt{e} \sim \mathtt{t}) \to \mathtt{t}) \sim \mathtt{t}) \to \mathtt{t}$.

(35) a. L everyone $(((\mathtt{e} \sim \mathtt{t}) \to \mathtt{t}) \sim B) \to B$ $\quad \lambda d.\, d(\lambda c.\, \forall x.\, cx)$
b. S L everyone $(((\mathtt{e} \sim B) \to B) \sim \mathtt{t}) \to \mathtt{t}$ $\quad \lambda d.\, \forall x.\, d(\lambda c.\, cx)$

We can understand how SKIP works by examining (35). LIFT adds the new continuation layer $((\cdots \sim B) \to B)$ on the outside, but SKIP LIFT inserts the new layer inside, promoting the original outer layer to the higher final level. Thus SKIP lets LIFT help *everyone* quantify at the second continuation level. In the semantic value of *S L everyone*, the universal quantifier is outside the second-order continuation variable *d*; this confirms that SKIP enables the quantifier introduced by *everyone* to skip over the first continuation level to take scope at the second level.

Replacing *L everyone* in (33) with *S L everyone* finally allows us to derive inverse scope as desired:

(36) Someone saw everyone.
E (S E (M (M B) (L someone) (M (M F) (L (L saw)) (S L everyone))))
$= \forall x.\, \exists y.\, \mathbf{saw}\, x\, y$

In general, quantifiers on the same level will take linear scope relative to one another, while quantifiers at higher levels take wider scope. Thus inverse scope is associated with extra effort (to the extent that category shifting can be thought of as correlating with processing effort), which correlates with the observation that, all else being equal, linear scope is more salient.[16]

[16] There are other derivational routes to the same result. For instance, it is possible for the second-order lifting to target the VP rather than the direct object: E (S E (S ((M B) someone) (S L (M F (L saw)) everyone))). This derivation is simpler in terms of number of type-shifting operators, though it is not necessarily easier to understand.

3.4. **Pronouns and quantificational binding.** Using the type-shifting operators introduced above, we can already handle quantificational binding just by adding a few lexical items. Crucially, the presence of these items in a clause affects its syntactic category and semantic type. As illustrated by (5) in Section 1.3, a pronoun changes the type of the containing expression so as to allow a binder access to the pronominal argument position. As discussed there, this type-directed strategy in general, and the specific choice of translating pronouns as the identity function in particular, closely resembles the variable-free program of Jacobson (e.g., 1999, 2000), and many of her discussions and arguments carry over here.

We will first contrast a pronoun with a proper name, then show how a quantifier can bind a pronoun.

(37) a. John left.
b. He left.

The sentence *John left* has category $\mathtt{t}$ and denotes the closed proposition that John left. The pronominal version, *He left*, is not closed: it depends for its value on the choice of an individual corresponding to the referent of *he*. On the standard treatment, this dependence is handled by relativizing the denotation of an expression to an assignment function that specifies the values of pronouns. In Jacobson's work and here, the open nature of the pronominal sentence is encoded more directly, in the category structure and the denotation of the expression itself: instead of denoting a proposition relative to an assignment, (37b) will denote a function from individuals to propositions. That is, the denotation of (37b) will be a function from possible choices for the value of *he* to the proposition that the chosen individual left.

Recall that our denotations for *everyone* and *someone* actively manipulate continuations in the sense that they map $\langle \mathtt{e}, \mathtt{t} \rangle$-continuations into $\mathtt{t}$-results—hence the category $(\mathtt{e} \sim \mathtt{t}) \to \mathtt{t}$. In all of our analyses so far, the category to the right of every '$\sim$' or '$\to$' connective is $\mathtt{t}$; this reflects the fact that all clauses considered so far denote propositions. For instance, the category $(\mathtt{e} \sim \mathtt{t}) \to \mathtt{t}$ requires a propositional type for its input argument ('$\sim \mathtt{t}$') and propagates it onward ('$\to \mathtt{t}$'). For a clause with an unbound pronoun like (37b), we want a denotation that maps an individual to a proposition. All we need to do to make this happen in the current system is to assign a denotation to pronouns that map continuations into not $\mathtt{t}$-results but $\langle \mathtt{e}, \mathtt{t} \rangle$-results.

But consider what the continuation of a pronoun is: it is a function from a normal NP meaning (type $\mathtt{e}$) to the type of meaning corresponding to a complete clause (type $\mathtt{t}$)—that is, it is a function from individuals to propositions, exactly what we are looking for. Thus what we need a pronoun to do is to make its own continuation the value of the containing expression. In other words, pronouns must denote the identity function on $\langle \mathtt{e}, \mathtt{t} \rangle$-continuations.

(38) a. John $\mathtt{e}$ $\quad \mathbf{j}$
b. L John $(\mathtt{e} \sim A) \to A$ $\quad \lambda c.\, c\mathbf{j}$
c. he $(\mathtt{e} \sim A) \to (\mathtt{e} \rhd A)$ $\quad \lambda c.\, c$

To contrast a pronoun with a proper name, let us consider *L John* and *he*, both first-order NP expressions. The first-order lifted version of *John* is of type $(\mathtt{e} \sim A) \to A$ as shown in (38b), which subsumes, for instance, $(\mathtt{e} \sim \mathtt{t}) \to \mathtt{t}$: a function from $\langle \mathtt{e}, \mathtt{t} \rangle$-continuations to truth values. A pronoun is also a function that takes $\langle \mathtt{e}, \mathtt{t} \rangle$-continuations as input, but its output has type $\langle \mathtt{e}, \mathtt{t} \rangle$ rather than $\mathtt{t}$.

(39) a. He left. $\quad \mathtt{e} \rhd \mathtt{t}$ $\quad \lambda x.\, \mathbf{left}\, x$
b. John saw him. $\quad \mathtt{e} \rhd \mathtt{t}$ $\quad \lambda x.\, \mathbf{saw}\, x\, \mathbf{j}$

c. Everyone saw him. $\mathtt{e} \rhd \mathtt{t}$ $\lambda x.\, \forall y.\, \textbf{saw}\; x\; y$

d. John thought she left. $\mathtt{e} \rhd \mathtt{t}$ $\lambda x.\, \textbf{thought}(\textbf{left}\; x)\; \textbf{j}$

The category of each of these sentences is $\mathtt{e} \rhd \mathtt{t}$, a function from a pronoun value to a proposition. Whether in subject or object position, whether embedded or not, pronouns find their way to the ultimate result type to add a layer of functional dependence to the denotation of the larger clause. We use the binary connective '$\rhd$' to record this dependence in the syntactic categories. The reason we use $\rhd$ rather than, say, $\sim$, is that a sentence containing a pronoun is not the same thing either conceptually or syntactically as a continuized sentence without a pronoun.

The ability of a deeply embedded pronoun to affect the result category of a larger expression that contains it is crucial to the way quantificational binding works in this system. (In addition, we shall see in Section 3.5 that *wh*-phrases and *wh*-traces use the same technique.) The fact that pronouns operate on continuations makes the value of that pronoun accessible to other expressions that manipulate continuations, in particular their binders. We introduce the following type-shift operator to make a noun phrase bind a pronoun:

(40) BIND $((\mathtt{e} \sim A) \to B) \Rightarrow ((\mathtt{e} \sim \mathtt{e} \rhd A) \to B)$ $\lambda X.\, \lambda c.\, X(\lambda x.\, cxx)$

We can then use BIND to type-shift *everyone* into a quantificational NP that binds as it quantifies:

(41) BIND everyone $= (\mathtt{e} \sim \mathtt{e} \rhd \mathtt{t}) \to \mathtt{t} : \lambda c.\, \forall x.\, cxx$

The normal *everyone* in (17) expects a continuation of type $\mathtt{e} \sim \mathtt{t}$ and returns a truth value. The binding version of *everyone*, computed in (41) as *BIND everyone*, also returns a truth value, but expects a continuation that has been augmented by a pronoun with an extra argument of type $\mathtt{e}$. The semantic value accomplishes this by feeding the individual chosen for each quantificational case to the augmented continuation twice: first as the entity standing in for the generalized quantifier, then again to bind the pronoun. This denotation embodies the fact that a quantifier can only bind a pronoun if it takes scope over that pronoun.

We can now derive *Everyone*$_i$ *thought he*$_i$ *left* and *Everyone*$_i$*'s mother thought he*$_i$ *left*:

(42) E (M B (BIND everyone) (M F (L thought) (M B he (L left))))
$= \forall x.\, \textbf{thought}\; (\textbf{left}\; x)\; x$

(43) E (M B (M B (BIND everyone's) (L mother)) (M F (L thought) (M B he (L left))))
$= \forall x.\, \textbf{thought}\; (\textbf{left}\; x)\; (\textbf{mother}\; x)$

Thus, just as in (26) and (27), the operators as given generalize to binding out of a possessive NP without further stipulation. Note in (43) that *everyone* need not c-command the pronoun to bind it. LF-based theories of binding typically need some principle which says in effect that if A can bind B, and A contains C, and C can take scope over B, then C can bind B (e.g., Ruys, 2000, p. 517). Since this principle is a theorem of our system, no such stipulation is required here.

Under this system, a binding quantifier and a pronoun must take effect at the same continuation level for binding to happen. (This fact is crucial for our account of crossover in Section 4.) Conversely, a quantifier binds a pronoun whenever they operate at the same level. Hence, if we restrict ourselves to first-order continuations, the only analysis available for the sentence *Everyone thought he left* is the reading in which *everyone* binds *he*. With the help of higher-order continuations, however, we can derive another reading in which *he* remains unbound: we simply apply S L to lift *he* to level 2 while (the non-binding version of) *everyone* continues to operate at level 1:

(44) E (S E (M (M B) (L everyone)
(M (M F) (L (L thought)) (M (M B) (S L he) (L (L left))))))
$= \lambda y.\, \forall x.\, \textbf{thought}\; (\textbf{left}\; y)\; x$

The final result is of category $\mathtt{e} \rhd \mathtt{t}$, as expected of any sentence with a single unbound pronoun. (Compare with the inverse scope derivation in (36).)

When there is more than one pronoun (e.g., *He saw her*), the resulting denotation will have two layers of functional dependence. In order for the two pronouns to keep out of each other's way, they must seek out different continuation levels. In effect, what most binding theories encode as an index on a pronoun here corresponds to a continuation level. As a result, we do not need to assume that pronouns are lexically polysemous: a single denotation will suffice, in combination with the independently needed category-shifting rules for scope manipulation. Similarly, we need only one denotation for a quantifier that binds a pronoun.

3.5. *Wh*-phrases, in-situ and raised.

To generate single and multiple *wh*-questions, we begin by providing the following lexical items:

(45) a. who $(\mathtt{e} \sim A) \to (\mathtt{e}\; ?\; A)$ **who**
b. what $(\mathtt{e} \sim A) \to (\mathtt{e}\; ?\; A)$ **what**

The syntactic category of *who* and *what* resembles that of *he* and *she*, with one difference: the type of the final result returned is $\mathtt{e}\; ?\; \mathtt{t}$ rather than $\mathtt{e} \rhd \mathtt{t}$. That is, a *wh*-word fits into a context expecting an NP (i.e., a context that produces a continuation of type $\mathtt{e} \sim A$ for some choice of A), and transforms that context into a question. The type $\mathtt{e}\; ?\; A$ is the type of a question in which an NP has been questioned, i.e., that "asks for" an individual. For example, we can derive a simple *wh*-question, a multiple *wh*-question, and a question in which the *wh*-word is embedded within a possessive as follows.

(46) Who left?
E (M B who (L left)) $= \mathtt{e}\; ?\; \mathtt{t} : \textbf{who}(\lambda x.\, \textbf{left}\; x)$

(47) Who bought what?
E (M B who ((S (F bought)) what)) $= \mathtt{e}\; ?\; (\mathtt{e}\; ?\; \mathtt{t}) : \textbf{who}(\lambda x.\, \textbf{what}(\lambda y.\, \textbf{bought}\; y\; x))$

(48) Whose mother left?
E (M B (M B who's (L mother)) (L left)) $= \mathtt{e}\; ?\; \mathtt{t} : \textbf{who}(\lambda x.\, \textbf{left}(\textbf{mother}\; x))$

Just as for quantificational possessors, the scoping mechanism allows *wh*-possessors embedded arbitrarily deeply (e.g., *whose mother's friend's dog left?*) to take effect at the top level of the sentence without special stipulation.

In order for a *wh*-phrase to occur in sentence-initial position (which we will inaccurately call "raised" in deference to universal usage), we now provide a silent pronoun ("trace") and a silent syntactic topicalization operator that allows a *wh*-phrase to occur in fronted position.

(49) a. _ $(\mathtt{e} \sim A) \to (\mathtt{e} \sim A)$ $\lambda c.\, c$
b. Q $(A \to (\mathtt{e}\; ?\; B)) \Rightarrow (A / (\mathtt{e}\; ?\; B))$ $\lambda X.\, X$

The trace _ is similar to the pronoun denotations given above in that it records on the result type the fact that an e-type argument is missing and needs to be connected with an additional argument in order to be complete.[17]

[17] We have provided a silent trace for ease of exposition, mainly in order to make it easy to compare the linear order of the argument position bound by the raised *wh*-phrase with any in-situ *wh*-phrase, but this is not an essential assumption. The analysis could easily be reconstructed in a system without any empty categories by

The topic operator Q converts a *wh*-phrase into the kind of phrase that can bind a trace. Because this operator is specific to *wh*-expressions, we call it Q, though in general, other types of expression can also topicalize in English. Assuming that *wondered* has type $(\texttt{e ? t})/\texttt{e}\backslash\texttt{t}$, we can now derive *Mary wondered what John bought.* Using the Q operator, *what* is able to bind the trace in final position. However, rather than constantly embedding questions under predicates like *wonder*, we will fake auxiliary inversion by adding an entry for *did* that has no syntactic nor semantic effect.

(50) did $\texttt{t/t}$ $\lambda p.\, p$

This provides for matrix questions containing one raised *wh*-phrase and any number of in-situ *wh*-phrases:

(51) What did Mary buy __?
F (Q what) (E (M F (L did) (M B (L Mary) (M F (L buy) _))))
$= \texttt{e ? t} : \textbf{what}(\lambda x.\, \textbf{buy}\ x\ \textbf{m})$

(52) Who did Mary give __ what?
F (Q who) (E (M F (L did) (M B (L Mary) (M F (M F (L give) _) what))))
$= \texttt{e ? (e ? t)} : \textbf{who}(\lambda x.\, \textbf{what}(\lambda y.\, \textbf{give}\ x\ y\ \textbf{m}))$

Interestingly, this analysis of English *wh*-phrases automatically rules out any sentence in which more than one *wh*-word has been fronted (e.g., **What who did Mary say __ bought __?*).

3.6. **Summary.** Here are all of the rules and lexical entries discussed above gathered together.

The composition rule:

Syntax:	$A \Rightarrow B$	followed by	A	yields	B
Semantics:	f		x		fx

Category-shifting rules:

F	$(A/B) \Rightarrow A \Rightarrow B$	$\lambda fx.\, fx$
B	$A \Rightarrow (A\backslash B) \Rightarrow B$	$\lambda xf.\, fx$
LIFT	$A \Rightarrow ((A \sim B) \to B)$	$\lambda xc.\, cx$
EVAL	$((\texttt{t} \sim \texttt{t}) \to A) \Rightarrow A$	$\lambda X.\, X(\lambda x.\, x)$
SKIP	$(A \Rightarrow B) \Rightarrow ((A \sim D) \to E) \Rightarrow ((B \sim D) \to E)$	$\lambda XYc.\, Y(\lambda y.\, c(Xy))$
META	$(A \Rightarrow B \Rightarrow C) \Rightarrow ((A \sim E) \to F) \Rightarrow ((B \sim D) \to E) \Rightarrow ((C \sim D) \to F)$	$\lambda XLRc.\, L(\lambda l.\, R(\lambda r.\, c(Xlr)))$
Q	$(A \to (\texttt{e} \mathrel{?} B)) \Rightarrow (A/(\texttt{e} \mathrel{?} B))$	$\lambda X.\, X$
BIND	$((\texttt{e} \sim A) \to B) \Rightarrow ((\texttt{e} \sim \texttt{e} \rhd A) \to B)$	$\lambda X.\, \lambda c.\, X(\lambda x.\, cxx)$

Lexical entries:

__ (trace)	$(\texttt{e} \sim A) \to (\texttt{e} \sim A)$	$\lambda c.\, c$	
everyone	$(\texttt{e} \sim \texttt{t}) \to \texttt{t}$	$\lambda c.\, \forall x.\, cx$	
someone	$(\texttt{e} \sim \texttt{t}) \to \texttt{t}$	$\lambda c.\, \exists x.\, cx$	
he	$(\texttt{e} \sim A) \to (\texttt{e} \rhd A)$	$\lambda c.\, c$	(similarly *him*, *his*)
who	$(\texttt{e} \sim A) \to (\texttt{e} \mathrel{?} A)$	**who**	(similarly *what*, *whose*)
Mary	$\texttt{e}$	**m**	
mother	$\texttt{e}\backslash\texttt{e}$	**mother**	
left	$\texttt{e}\backslash\texttt{t}$	**left**	

replacing trace with a category-shifting operator with category $(e|A) \Rightarrow ((A \sim B) \to (e \sim B))$ (where '|' generalizes over \ and /) and denotation $\lambda Acx.\, c(Ax)$.

did	$\texttt{t/t}$	$\lambda p.\, p$
saw	$\texttt{e/e}\backslash\texttt{t}$	**saw**
gave	$\texttt{e/e/e}\backslash\texttt{t}$	**gave**
thought	$\texttt{t/e}\backslash\texttt{t}$	**thought**
wondered	$(\texttt{e ? t})/\texttt{e}\backslash\texttt{t}$	**wondered**

When assessing the complexity of this fragment, bear in mind what is not present: there is no quantifier movement, no LF, no assignment functions, and no constraints on the application of the category-shifting rules. Nevertheless, this fragment provides a reasonably robust account of quantifier scope, quantificational binding, binding out of NP, and single and multiple *wh*-question formation.

Two features of the grammar that are especially appealing are that it allows inverse linking, quantifier scope, and binding out of an NP without any special rules or constraints; and that *wh*-words receive a single denotation that serves equally well in-situ as well as raised.

Obviously, there are many intricate details of quantification, binding, and *wh*-phrases that are not addressed here, since we have provided only what is necessary to consider crossover and superiority.

Crucially, nothing in the fragment is specific to either crossover or superiority: in particular, the category-shifting rules are as simple as they can be and still provide a general treatment of scope, and the denotations for quantifiers, pronouns, and *wh*-words are as general as they can be and still provide a reasonable approximation of their contribution to truth conditions. Nevertheless, we are about to see that crossover and superiority fall out automatically from this fragment without any addition stipulation.

4. Explaining Crossover

Now we are ready to show how the system accounts for crossover. We will compare the following two examples:

(53) a. Everyone$_i$ saw his$_i$ mother.
b. *His$_i$ mother saw everyone$_i$.

The grammatical example (53a) is perfectly straightforward to derive:

(54) E (M B (BIND everyone) (M F (L saw) (M B his (L mother))))
$= \forall x.\, \textbf{saw}\ (\textbf{mother}\ x)\ x$

The crossover-violating example (53b), however, has no analysis (at least, none whose result has type $\texttt{t}$). The reason is as follows. In this fragment (as in every coherent binding system), a quantifier must take scope over a pronoun in order to bind it. For a quantifier to bind a preceding pronoun, it must lift to a higher continuation level in order to take inverse scope over that pronoun. But the quantifier can only bind a pronoun that is at the same level as the quantifier, as we mentioned in Section 3.4. Because the quantifier cannot operate both at the same level as the pronoun and at a higher level, the system correctly fails to generate (53b).

Furthermore, because our account predicts that binding will be possible whenever a quantifier takes scope over a pronoun that follows it, we can generate examples in which the quantificational NP fails to c-command the bound pronoun in surface structure. In addition to standard examples of binding out of DP such as (43), our analysis generalizes to other configurations of the *We will sell no wine before its time* type. To illustrate, we will consider a case of binding into an adjunct:

(55) Mary phoned everyone$_i$ on his$_i$ birthday.
E (M B (L Mary) (M B (M F (L phoned) (BIND everyone))
(M F (L on) (M B his (L birthday)))))
$= \forall x.\, \textbf{on}\,(\textbf{birthday}\ x)\,(\textbf{saw}\ x)\,\textbf{m}$

All we need is a basic lexical entry for adjunct-creating prepositions; here, we assume that *on* is of category e/(e\t)\e\t.

Examples like (55), in which a quantifier embedded within a verb phrase can bind into a following adjunct, are problematic for LF-style theories on which binding depends on c-command at surface structure (e.g., the theory in Büring, 2001). Our theory not only generates these cases as desired, but also correctly predicts that reversing the binder and pronoun positions results in a crossover violation, e.g., **Mary phoned his$_i$ mother on everyone$_i$'s birthday*.

5. Explaining Superiority

Just as for quantificational binding and crossover, nothing more need be said in order to account for superiority contrasts. However, understanding the fragment's failure to generate superiority violations requires some explanation.

First, let us examine a grammatical multiple *wh*-question, and then try to understand why a superiority violation does not go through. Consider the grammatical (52), repeated below.

(52) Who did Mary give __ what?
F (Q who) (E (M F (L did) (M B (L Mary) (M F (M F (L give) _) what))))
$= \mathtt{e}\ ?\ (\mathtt{e}\ ?\ \mathtt{t}) : \textbf{who}(\lambda x.\, \textbf{what}(\lambda y.\, \textbf{give}\ x\ y\ \textbf{m}))$

In the derivation (52), the trace scopes over the in-situ *wh*-phrase *what*. This scope relationship is evident from the fact that the subexpression *did Mary give __ what* has the category $\mathtt{e} \sim \mathtt{e}\ ?\mathtt{t}$, in which the outer connective ~ is contributed by the trace, and the inner connective ? is contributed by *what*. Accordingly, the denotation of that subexpression is $\lambda x.\, \textbf{what}(\lambda y.\, \textbf{give}\ x\ y\ \textbf{m})$ rather than $\textbf{what}(\lambda y.\, \lambda x.\, \textbf{give}\ x\ y\ \textbf{m})$. This scope relationship is crucial for the raised *wh*-phrase *who* to enter the derivation, since *Q who* expects an argument whose outer connective is ~. In general, a *wh*-trace must take scope over any in-situ *wh*-phrases for the same clause, in order for its corresponding raised *wh*-phrase to enter the derivation.

Now consider the superiority-violating counterpart of (52).

(56) *What did Mary give who __?

This sentence cannot be a member of the double-*wh* syntactic category $\mathtt{e}\ ?\ \mathtt{e}\ ?\ \mathtt{t}$, because the subexpression *did Mary give who __* cannot be a member of the category $\mathtt{e} \sim \mathtt{e}\ ?\ \mathtt{t}$, in which the trace takes inverse scope over the in-situ *wh*-phrase *who*. Let us try to create such an inverse scope derivation, following the same strategy that gave rise to quantifier scope ambiguity in Section 3.3, and see what blocks our attempt. To begin, we may make the first-order trace operate at the second continuation level by lifting it with SKIP LIFT, as shown below.

(57) M (M F) (L (L did)) (M (M B) (L (L Mary))
(M (M F) (M (M F) (L (L give)) (L who)) (S L _)))
$= (((\mathtt{t} \sim A) \rightarrow (\mathtt{e}\ ?\ A)) \sim B) \rightarrow (\mathtt{e} \sim B) : \lambda c.\, \lambda y.\, c(\lambda d.\, \lambda x.\, d(\textbf{give}\ x\ y\ \textbf{m}))$

So far so good: the trace is operating at the second level of continuations $((\cdots \sim B) \rightarrow (\mathtt{e} \sim B))$, outside the first level, where the in-situ *who* is operating $((\cdots \sim A) \rightarrow (\mathtt{e}\ ?\ A))$.

Because the result of (57) contains two continuation levels, we must lower it twice before combining it with the raised *wh*-phrase *what*. First, we apply SKIP EVAL; this requires that the category A be equal to t.

(58) $\text{S E (57)} = ((\mathtt{e}\ ?\ \mathtt{t}) \sim B) \rightarrow (\mathtt{e} \sim B) : \lambda c.\, \lambda y.\, c(\lambda x.\, \textbf{give}\ x\ y\ \textbf{m})$

Next, we would like to apply EVAL, but we cannot, because that would require that the category e ? t be equal to t. Because EVAL as specified in (16b) requires the category t, a trace cannot outscope a linearly preceding in-situ *wh*-phrase in a complete derivation.[18] That is why superiority violations such as (56) are not generated by the fragment.

5.1. An empirical refinement: D-linked *wh*-phrases.

Since Pesetsky's (1987) work, it is usually assumed that if an in-situ *wh*-phrase is somehow linked to the discourse (what it means to be 'linked to the discourse' currently remains imprecise), it no longer triggers superiority violations.

(59) What did which students buy __?

This would normally be a superiority violation, since *what* crosses over the in-situ *wh*-phrase *which students*. But *wh*-phrases headed by *which* are assumed to be intrinsically linked to the discourse (D-linked) by virtue of some aspect of their meaning, which is supposed to be why sentences like (59) are grammatical.

So if we want to derive (59), we must find a way for the in-situ *wh*-phrase to allow the trace to take wide scope. This can be accomplished by allowing the *which*-phrase to explicitly anticipate the possibility that a trace may follow it:

(60)	what	$(\mathtt{e} \sim A) \rightarrow (\mathtt{e}\ ?\ A)$	**what**
(61)	which N	$(\mathtt{e} \sim (\mathtt{e} \sim A)) \rightarrow (\mathtt{e} \sim (\mathtt{e}\ ?\ A))$	$\lambda c.\, \lambda t.\, \textbf{which}(\textbf{N})(\lambda x.\, cxt)$

In the non-D-linked denotation for *what* (repeated here from (45b) for comparison), there is one 'e~'; in the D-linked denotation for a *wh*-phrase headed by *which*, there is an extra 'e~' corresponding to the downstream trace. The semantics simply gives the argument corresponding to the trace (here, t) wide scope by fiat.

This allows the following derivation for (59):

(62) F (Q what) (E (M B which-students (M F (L buy) _)))
$= \mathtt{e}\ ?\ (\mathtt{e}\ ?\ \mathtt{t}) : \textbf{what}(\lambda t.\, \textbf{which}(\textbf{students})(\lambda x.\, \textbf{buy}\ t\ x))$

Needless to say, there are additional complexities related to D-linking that we are not able to address here.

6. Wh-binding and Crossover

As is well-known, *wh*-phrases are capable of binding pronouns, and when they do, they exhibit crossover effects—but with an interesting twist:

(63) a. Who$_i$ __ saw his$_i$ mother?
b. *Who$_i$ did his$_i$ mother see __?

[18] To make more precise what we mean by a 'complete derivation', say that a derivation of a sentence or a question is 'complete' if the expression as a whole is at the zeroth continuation level. For practical purposes, this will mean that the final type contains no occurrences of the connective '~'. Thus the main complete categories provided by the grammar are t, $\mathtt{e} \rhd \mathtt{t}$, e ? t, e ? (e ? t), and so on. For instance, the grammar classifies the sentence *Mary left* as belonging both to the category t and the category $(\mathtt{t} \sim \mathtt{t}) \rightarrow \mathtt{t}$; but since $(\mathtt{t} \sim \mathtt{t}) \rightarrow \mathtt{t}$ is at the first continuation level (it contains a ~), only the derivation that justifies inclusion in the category t counts as complete.

If the *wh*-phrase is doing the binding, then the pattern here seems mysterious at first glance, since in both cases the raised *wh*-phrase is to the left of the pronoun in question, c-commands it, and outscopes it.

As a first approximation, Reinhart (1983) (p. 114) and others have suggested that it is the *wh*-trace that binds the pronoun, and not the raised *wh*-word directly. In our system, this corresponds to applying BIND operator to the *wh*-trace:

(64) BIND __ = $(\mathtt{e} \sim (\mathtt{e} \rhd A)) \to (\mathtt{e} \sim A) : \lambda cx.\, cxx$

This is sufficient to derive the grammatical example in (63a):

(65) F (Q who) (E (M B (BIND __) (M F (L saw) (M B his (L mother)))))
$= \mathtt{e} \mathbin{?} \mathtt{t} : \textbf{who}(\lambda x.\, \textbf{saw}(\textbf{mother}\ x)\ x)$

The ungrammatical sentence in (63b) has no analogous derivation. The reason is that in order for the trace to serve as a binder, it must be evaluated before the pronoun, which means that it must precede the pronoun.

However, merely allowing the trace to bind is not sufficient to generate the full range of grammatical interpretations:

(66) a. Whose$_i$ friend$_j$'s next-door neighbor$_k$ did Mary think __ saw his$_{i/j/k}$ mother?
b. Whose$_i$ friend$_j$'s next-door neighbor$_k$ did Mary think his$_{*i/*j/*k}$ mother saw __?

In (66a), the person doing the seeing (i.e., the trace) must be a next-door neighbor, and this is guaranteed by our analysis. The problem is that if the pronoun is bound by the trace, we should expect that the pronoun also can refer only to the neighbor; but in fact, as indicated by the subscripts, it can be bound either by *whose*, or by *whose friend*, or by *whose friend's next-door neighbor.* Apparently (contra Reinhart's suggestion), in some situations, at least, the pronoun can be bound directly by a *wh*-phrase. Nevertheless, the trace still plays a critical role, as shown in (66b), since if the trace fails to precede the pronoun, none of these binding possibilities are possible.

Remarkably, this intricate pattern of facts falls out here without stipulation. Our system does in fact allow a *wh*-phrase to bind the pronoun directly. For instance, the sentence derived above in (65) by allowing the trace to bind the pronoun can also be derived as follows:

(67) F (Q (BIND who)) (E (M B __ (M F (L saw) (M B his (L mother)))))
$= \mathtt{e} \mathbin{?} \mathtt{t} : \textbf{who}(\lambda x.\, \textbf{saw}(\textbf{mother}\ x)\ x)$

The variability in the binding shown in (66a) arises from the variability of applying the BIND operator to either *whose*, *whose friend*, or *whose friend's next-door neighbor.* We illustrate this interaction of binding with pied-piping with the following simpler example:

(68) Whose$_i$ friend __ saw his$_i$ mother?
F (Q (M B (BIND whose) (L friend)))
(E (M B __ (M F (L saw) (M B his (L mother)))))
$= \mathtt{e} \mathbin{?} \mathtt{t} : \textbf{who}(\lambda x.\, \textbf{saw}(\textbf{mother}\ x)\ (\textbf{friend}\ x))$

As desired, the trace corresponds to the friend participant, but the pronoun is bound by *whose*.

Crucially, even considering the possibility that *wh*-phrases can bind pronouns directly, the ungrammatical interpretations in (63b) and (66b) still have no derivation. The reason is that (BIND who) has category $(\mathtt{e} \sim (\mathtt{e} \rhd A)) \to \mathtt{e} \mathbin{?} A$. Note that the $\mathtt{e}$ corresponding to the trace is outside the $\mathtt{e}$ corresponding to the pronoun (i.e., the ~ connective is outside the ⊳ connective). This means that the trace must always outscope the pronoun, and by reasoning similar to the explanation for superiority, it follows that the trace must precede the pronoun.

A final detail relating to the interaction of *wh*-fronting with binding: as explained above in Section 4, a quantifier embedded in a verb phrase can bind into an adjunct, and, as noted by Lasnik and Stowell (1991), analogous facts hold for binding by a *wh*-trace:

(69) Which actress$_i$ did you [admit you had an affair with $\_\__i$] [after she$_i$ died]?

The point is that the adjunct containing the bound pronoun modifies the entire complex verb phrase in which the *wh*-trace is embedded. After all, presumably it is the admitting event and not the affair that takes place after the dying event (thanks to Daniel Büring for this example). We derive a less deeply embedded example that makes the same point:

(70) Who$_i$ did Mary call $\_\__i$ on his$_i$ birthday?
F (Q who) (E (S E (S (M B (L Mary)) (S L (M B (M F (L call) (BIND __))
(M F (L on) (M B his (L birthday))))))))
$= \mathtt{e} \mathbin{?} \mathtt{t} : \textbf{who}(\lambda x.\, \textbf{on}(\textbf{birthday}\ x)(\textbf{call}\ x)\ m)$

In the same way that a quantifier embedded in a verb phrase can take scope over and bind a pronoun in a following adjunct, the binding version of the trace can in effect take scope over and bind a pronoun to its right. Once again, examples like these are problematic for theories that attempt to derive binding from surface c-command relations, but fall out without stipulation here.

7. Evaluation Order and the META Rule

What precisely is the connection between order of evaluation and crossover and superiority?

We have claimed that the fragment has a left-to-right bias built in, but it may not be obvious where exactly in the grammar that bias resides. For instance, it might seem as if the locus of the directionality is in the combination rule (11), since that rule requires that when two phrases combine, the functor must be on the left; but in fact, evaluation order has nothing to do with the distinction between functors and arguments. After all, in the sentence *Mary left*, it is the argument *Mary* that occurs to the left of (and is evaluated before) the functor *left*, thanks to the category-shifting rule B. Another way of putting it is that the sentence *Mary left* must denote **left m** whether *Mary* is evaluated first or *left* is. Thus the relevant notion of the direction of evaluation is independent of the direction of functional application.

Rather, order of evaluation has to do with composition of higher-order values. In other words, order of evaluation only makes a difference in the presence of expressions that explicitly manipulate continuations, that is, whose lexical denotations contain the connective ~. Examination of the grammar and derivations above will reveal that the only way to combine expressions at higher continuation levels is with (some operator derived from) the META rule M. M is a recipe for how to put together two higher-level expressions; and in fact, M is the true locus of the evaluation bias.

7.1. M versus W in English.

Perhaps the best way to understand the role of M in determining order of evaluation is to consider the following alternative version of the META rule, which differs only in that it evaluates expressions right to left instead of left to right. We'll call the alternative operator W (an 'M' turned upside down):

(71) a. Left-to-right META (M) [same as (16d)]:

$$(A \Rightarrow B \Rightarrow C) \Rightarrow ((A \sim E) \to F) \Rightarrow ((B \sim D) \to E) \Rightarrow ((C \sim D) \to F)$$

Semantics: $\lambda XLRc.\, L(\lambda l.\, R(\lambda r.\, c(Xlr)))$

b. Right-to-left META (W):

$$(A \Rightarrow B \Rightarrow C) \Rightarrow ((A \sim D) \to E) \Rightarrow ((B \sim E) \to F) \Rightarrow ((C \sim D) \to F)$$

Semantics: $\lambda XLRc.\, R(\lambda r.\, L(\lambda l.\, c(Xlr)))$

There are subtle changes in the categories, but the clearest place to discern the change in evaluation direction is in the semantics: in the left-to-right version, the first, leftmost argument L has in view all of the information contained in R. In the right-to-left version, R has L in view, and the scope relations are reversed.

If we replace M with W in our grammar, the order of evaluation switches from left-to-right to right-to-left. Consequently, we should expect a corresponding reversal in the behavior of phenomena that are sensitive to order of evaluation. And this is in fact exactly what happens. We have discussed three order-sensitive phenomena: quantifier scope relations, crossover, and superiority, so let us consider each of these in turn.

First, as we saw in (28) (repeated here), M produces a bias in favor of linear scope in the following sense: on the simplest derivation, quantificational expressions on the left outscope those to the right.

(72) Someone saw everyone.
E (M B someone (M F (L saw) everyone)) = $\exists y.\, \forall x.\, \textbf{saw}\, y\, x$

(73) Someone saw everyone.
E (W B someone (W F (L saw) everyone)) = $\forall x.\, \exists y.\, \textbf{saw}\, y\, x$

In (72), M is used, and the default scope is left-to-right. But if W is used, as in (73), default scope relations are reversed, and there is a right-to-left bias in quantificational scoping.[19]

Second, if we replace M with W in our grammar, then the crossover-violation example (53b) becomes easy to derive:

(74) *His$_i$ mother saw everyone$_i$.
E (W B (W B his (L mother)) (W F (L saw) (BIND everyone)))
= $\forall x.\, \textbf{saw}\, x\, (\textbf{mother}\, x)$

By the reasoning given above in Section 4, the quantifier must be evaluated before the pronoun it binds; but since the order of evaluation is reversed when using W, the quantifier gets evaluated first, and the ungrammatical crossover violation is incorrectly generated. At the same time, the grammatical sentence (53a) is no longer generated, for reasons exactly analogous to the explanation in Section 4.

Finally, we can convince ourselves that evaluation order plays a crucial part in the explanation for superiority. With W in play, superiority predictions are reversed: the normally grammatical *Who did Tom say bought what?* fails to be generated, and the superiority violation can be derived as follows:

(75) *What did who buy __?
F (Q what) (E (W B who (W F (L buy) _))) = `e?(e?t)` : $\textbf{what}(\lambda y.\, \textbf{who}(\lambda x.\, \textbf{buy} y x))$

In this derivation (and in (74)) it remains true that the *wh*-trace must be evaluated before any in-situ *wh*-phrase. For left-to-right evaluation (using M), that means that the *wh*-trace must precede the in-situ *wh*-phrases; but for right-to-left evaluation (using W), the *wh*-trace must follow the in-situ *wh*-phrases, since expressions on the right get evaluated first.

[19] Barker (2002a) in effect allows both M and W in a single grammar. This permits quantifier scope ambiguity without resorting to higher-order continuations.

We can now make precise our proposal that English evaluates expressions from left to right: this is nothing more than stipulating that the grammar contains M but not W. Such a grammar embodies a bias for linear scope, yet still allows inverse scope; allows quantificational binding, but not crossover; and generates multiple *wh*-questions, but not superiority violations.

Incidentally, unlike some types of ungrammaticality, crossover violations and superiority violations remain interpretable with enough mental effort, and violations are often judged as not so bad, especially in comparison with, e.g., strong crossover. In the context of our explanation here, we interpret this as suggesting that speakers are capable of resorting to a non-default processing strategy (temporarily using W instead of M) given sufficient motivation.

Thus our predictions of crossover and superiority both depend on the order of evaluation, and for both phenomena, reversing the order of evaluation reverses the patten of predicted grammaticality. This is the sense in which left-to-right evaluation provides a unified account of scope, crossover and superiority.

7.2. **M versus W in programming languages.** When we say that M and W differ in evaluation order, we are using the notion of evaluation order that is standard in programming language research (Meyer and Wand, 1985, p. 223; Danvy and Filinski, 1989, p. 15; Papaspyrou, 1998). To illustrate this connection explicitly, let us analyze the evaluation order of a simple programming language using continuations.

Consider the following language of arithmetic expressions: an expression is either a number or two expressions conjoined with a plus sign `+`. Each expression can be *evaluated* to give a numeric result.

(76)

	Expression	*Result*
a.	`2`	2
b.	`2 + 3`	5

Suppose that our computer is connected to a printer. To use the printer, we add a new feature `print` to our programming language. If *e* is an expression, then `print(`*e*`)` is also an expression. The expression `print(`*e*`)` evaluates just as *e* does, but in addition sends the result to the printer.

(77)

	Expression	*Result*	*Printer*
a.	`2 + 3`	5	(no output)
b.	`print(2 + 3)`	5	5
c.	`2 + print(3)`	5	3

We have yet to specify what happens if a single expression invokes `print` more than once. For instance, suppose we evaluate `print(2) + print(3)`. The numeric result produced should clearly be 5, but it is unclear whether the printer should print 2 followed by 3 or vice versa. Intuitively, if the subexpression `print(2)` is evaluated first, then 2 will be printed first.

The behavior of this programming language can be modeled using continuations. Let `n` be the type of numbers, and `p` be the type of printer outputs. (Perhaps each `p`-value is a finite sequence of `n`-values.) Every expression denotes something of the type $\langle\langle n, p\rangle, p\rangle$. The semantic rules are as follows.

(78) a. $[\![2]\!] = \lambda c.\, c(2)$

b. $\llbracket 3 \rrbracket = \lambda c.\, c(3)$

c. i. $\llbracket e_1 \texttt{ + } e_2 \rrbracket = \lambda c.\, \llbracket e_1 \rrbracket(\lambda n_1.\, \llbracket e_2 \rrbracket(\lambda n_2.\, c(n_1 + n_2)))$ left to right

ii. $\llbracket e_1 \texttt{ + } e_2 \rrbracket = \lambda c.\, \llbracket e_2 \rrbracket(\lambda n_2.\, \llbracket e_1 \rrbracket(\lambda n_1.\, c(n_1 + n_2)))$ right to left

d. $\llbracket \texttt{print}(e) \rrbracket = \lambda c.\, \llbracket e \rrbracket(\lambda n.\, \mathbf{cons}(n, c(n)))$

The rule (78c) above gives two alternative semantic rules for expressions built with the plus sign `+`. The first alternative (78ci) implements left-to-right evaluation and is analogous to M; the second (78cii) implements right-to-left evaluation and is analogous to W. Also, the rule (78d) uses an auxiliary two-place function **cons**, defined as follows: if n is a number and p is a piece of printer output, then $\mathbf{cons}(p, n)$ is another piece of printer output, the result of prepending n to p.

Given any expression e, we can use the semantic rules in (78) to compute the printer output generated by e: it is $\llbracket e \rrbracket(\lambda n.\, \mathbf{nil})$, where $\lambda n.\, \mathbf{nil}$ is the constant function returning **nil**, the empty output. For example, if we pick the left-to-right rule (78ci) and disregard the right-to-left rule (78cii), then the denotation of `print(2) + print(3)` is

$$
\begin{aligned}
&\llbracket \texttt{print(2) + print(3)} \rrbracket \\
&\quad = \lambda c.\, \llbracket \texttt{print(2)} \rrbracket(\lambda n_1.\, \llbracket \texttt{print(3)} \rrbracket(\lambda n_2.\, c(n_1 + n_2))) \\
&\quad = \lambda c.\, (\lambda c.\, \llbracket 2 \rrbracket(\lambda n.\, \mathbf{cons}(n, c(n)))) \\
&\qquad\quad (\lambda n_1.\, (\lambda c.\, \llbracket 3 \rrbracket(\lambda n.\, \mathbf{cons}(n, c(n))))(\lambda n_2.\, c(n_1 + n_2))) \\
&\quad = \lambda c.\, (\lambda c.\, (\lambda c.\, c(2))(\lambda n.\, \mathbf{cons}(n, c(n)))) \\
&\qquad\quad (\lambda n_1.\, (\lambda c.\, (\lambda c.\, c(3))(\lambda n.\, \mathbf{cons}(n, c(n))))(\lambda n_2.\, c(n_1 + n_2))) \\
&\quad = \lambda c.\, (\lambda c.\, \mathbf{cons}(2, c(2)))(\lambda n_1.\, (\lambda c.\, \mathbf{cons}(3, c(3)))(\lambda n_2.\, c(n_1 + n_2))) \\
&\quad = \lambda c.\, \mathbf{cons}(2, \mathbf{cons}(3, c(5))),
\end{aligned}
$$

so the printer output generated is

$$(\lambda c.\, \mathbf{cons}(2, \mathbf{cons}(3, c(5))))(\lambda n.\, \mathbf{nil}) = \mathbf{cons}(2, \mathbf{cons}(3, \mathbf{nil})),$$

in which 2 precedes 3. If we pick the right-to-left rule (78cii) rather than the left-to-right rule (78ci), then 3 would precede 2 in the printer output.

8. Conclusions

We have presented a concrete analysis that explains how the exact configurations that violate crossover and superiority follow directly from the assumption that expressions are evaluated from left to right: when the type-shifting rule M is replaced with W, the pattern of configurations predicted to violate crossover and superiority is reversed.

There can be little doubt that language has a bias for evaluating expressions in the order in which they are heard, at least at some level. Nevertheless, there are many different ways in which this general principle could be built into a grammar. We have provided one specific hypothesis giving a precise picture of what it means for a language to evaluate expressions from left to right. Furthermore, our notion of order of evaluation is the same notion proposed independently for characterizing the evaluation disciplines of computer programming languages. Perhaps surprisingly, stipulating that processing proceeds left to right automatically predicts crossover and superiority without any additional stipulation beyond providing the expressions involved with suitable denotations.

In computer science, continuations are a prime tool for analyzing execution disciplines (order of evaluation, call-by-value versus call-by-name, etc.) and side effects (printing, input, errors, etc.). In natural language, we have suggested that two fairly mysterious phenomena, crossover and superiority, arise from a bias in natural language for left-to-right evaluation. Whether or not our specific version of the left-to-right processing hypothesis is correct, making the hypothesis precise and exploring its predictions requires the use of continuations as an analytic tool. This suggests that continuations have a valuable role to play in the search for new and deeper insights into the structure of language.

References

Baker, Carl Leroy. 1970. Notes on the description of English questions: The role of an abstract question morpheme. *Foundations of Language* 6:197–219.

Barker, Chris. 2001. Continuations: In-situ quantification without storage or type-shifting. In Hastings et al. (2001).

———. 2002a. Continuations and the nature of quantification. *Natural Language Semantics* To appear.

———. 2002b. Remark on Jacobson 1999: Crossover as a local constraint. *Linguistics and Philosophy* To appear.

Büring, Daniel. 2001. A situation semantics for binding out of DP. In Hastings et al. (2001).

Chierchia, Gennaro. 1991. Functional WH and weak crossover. In *Proceedings of the 10th West Coast Conference on Formal Linguistics*, ed. Dawn Bates, 75–90. Stanford, CA: Center for the Study of Language and Information.

———. 1993. Questions with quantifiers. *Natural Language Semantics* 1(2):181–234.

Chomsky, Noam. 1973. Conditions on transformations. In *A festschrift for Morris Halle*, ed. Stephen Anderson and Paul Kiparsky, 232–286. New York: Holt, Rinehart and Winstron.

Comorovski, I. 1996. *Interrogative phrases and the syntax-semantics interface*. Dordrecht: Kluwer.

Danvy, Olivier, and Andrzej Filinski. 1989. A functional abstraction of typed contexts. Tech. Rep. 89/12, DIKU, University of Copenhagen, Denmark. `http://www.daimi.au.dk/~danvy/Papers/fatc.ps.gz`.

———. 1990. Abstracting control. In *Proceedings of the 1990 ACM conference on Lisp and functional programming*, 151–160. New York: ACM Press.

Dayal, Veneeta. 1996. *Locality in wh quantification: Questions and relative clauses in hindi*. Dordrecht: Kluwer.

———. 2003. Multiple wh questions. In *The Blackwell companion to syntax*, ed. Martin Everaert and Henk C. van Riemsdijk. Oxford: Blackwell.

Dowty, David. 1991. 'Variable-free' syntax, variable-binding syntax, the natural deduction lambek calculus, and the crossover constraint. In *Proceedings of the 11th West Coast Conference on Formal Linguistics*, ed. Jonathan Mead. Stanford, CA: Center for the Study of Language and Information.

de Groote, Philippe. 2001. Type raising, continuations, and classical logic. In *Proceedings of the 13th Amsterdam Colloquium*, ed. Robert van Rooy and Martin Stokhof, 97–101. Institute for Logic, Language and Computation, Universiteit van Amsterdam.

Hastings, Rachel, Brendan Jackson, and Zsofia Zvolensky, eds. 2001. *SALT XI: Semantics and linguistic theory*. Ithaca: Cornell University Press.

Heim, Irene, and Angelika Kratzer. 1998. *Semantics in generative grammar*. Oxford: Blackwell.

Hendriks, Herman. 1988. Type change in semantics: The scope of quantification and coordination. In *Categories, polymorphism and unification*, ed. Ewan Klein and Johan van Benthem, 96–119. Centre for Cognitive Science, University of Edinburgh.

———. 1993. Studied flexibility: Categories and types in syntax and semantics. Ph.D. thesis, Institute for Logic, Language and Computation, Universiteit van Amsterdam.

Higginbotham, James. 1980. Pronouns and bound variables. *Linguistic Inquiry* 11:679–708.

Hindley, J. Roger, and Jonathan P. Seldin. 1986. *Introduction to combinators and λ-calculus*. Cambridge: Cambridge University Press.

Hornstein, Norbert. 1995. *Logical form: From GB to Minimalism*. Oxford: Blackwell.

Jacobson, Pauline. 1999. Towards a variable-free semantics. *Linguistics and Philosophy* 22(2):117–184.

———. 2000. Paycheck pronouns, Bach-Peters sentences, and variable-free semantics. *Natural Language Semantics* 8(2):77–155.

Kuno, Susumu, and Soo-Yeon Kim. 1994. The weak crossover phenomena in Japanese and Korean. In *Japanese/Korean Linguistics*, ed. Noriko Akatsuka, vol. 4, 1–38. Stanford, CA: Center for the Study of Language and Information.

Kuno, Susumu, and Jane J. Robinson. 1972. Multiple wh questions. *Linguistic Inquiry* 3: 463–487.

Lasnik, Howard, and Tim Stowell. 1991. Weakest crossover. *Linguistic Inquiry* 22:687–720.

Meyer, Albert R., and Mitchell Wand. 1985. Continuation semantics in typed lambda-calculi (summary). In *Logics of programs*, ed. Rohit Parikh, 219–224. Lecture Notes in Computer Science 193, Berlin: Springer-Verlag.

Montague, Richard. 1974. The proper treatment of quantification in ordinary English. In *Formal philosophy: Selected papers of Richard Montague*, ed. Richmond Thomason, 247–270. New Haven: Yale University Press.

O'Neil, John. 1993. A unified analysis of superiority, crossover, and scope. In *Harvard working papers in linguistics*, ed. Höskuldur Thráinsson, Samuel D. Epstein, and Susumu Kuno, vol. 3, 128–136. Cambridge: Harvard University.

Papaspyrou, Nikolaos S. 1998. Denotational semantics of evaluation order in expressions with side effects. In *Recent advances in information science and technology: 2nd part of the proceedings of the 2nd IMACS international conference on circuits, systems and computers*, ed. Nikos E. Mastorakis, 87–94. Singapore: World Scientific.

Partee, Barbara H. 1987. Noun phrase interpretation and type-shifting principles. In *Studies in discourse representation theory and the theory of generalized quantifiers*, ed. Jeroen Groenendijk, Dick de Jongh, and Martin Stokhof, 115–143. Groningen-Amsterdam Studies in Semantics 8, Dordrecht: Foris.

Partee, Barbara H., and Mats Rooth. 1983. Generalized conjunction and type ambiguity. In *Meaning, use and interpretation of language*, ed. Rainer Bäuerle, Christoph Schwarze, and Arnim von Stechow, 361–383. Berlin: de Gruyter.

Pesetsky, David. 1982. Paths and categories. Ph.D. thesis, Department of Linguistics and Philosophy, Massachusetts Institute of Technology.

———. 1987. Wh-in-situ: Movement and unselective binding. In *The representation of (in)definiteness*. Cambridge: MIT Press.

Postal, Paul M. 1993. Remarks on weak crossover effects. *Linguistic Inquiry* 24:539–556.

Postal, Paul Martin. 1971. *Cross-over phenomena*. New York: Holt, Rinehart and Winston.

Potts, Christopher. 2001. (only) some crossover effects repaired. *Snippets* 3:13–14.

Reinhart, Tanya. 1983. *Anaphora and semantic interpretation*. London: Croom Helm.

Ruys, E.G. 2000. Weak crossover as a scope phenomenon. *Linguistic Inquiry* 31:513–539.

Shan, Chung-chieh. 2002. A continuation semantics of interrogatives that accounts for Baker's ambiguity. In *SALT XII: Semantics and linguistic theory*, ed. Brendan Jackson, 246–265. Ithaca: Cornell University Press.

Steedman, Mark. 2000. *The syntactic process*. MIT Press.

Tanenhaus, M. K., S. M. Garnsey, and J. Boland. 1990. Combinatory lexical information and language comprehension. In *Cognitive models of speech processing: Psycholinguistic and computational perspectives*, ed. G. T. M. Altmann, 383–408. Cambridge: MIT Press.

Division of Engineering and Applied Sciences, Harvard University
33 Oxford Street, Cambridge, MA 02138, USA
E-mail address: `ccshan@post.harvard.edu`
URL: `http://www.digitas.harvard.edu/~ken/`

0108 Department of Linguistics, University of California, San Diego
9500 Gilman Drive, La Jolla, CA 92093, USA
E-mail address: `barker@ucsd.edu`
URL: `http://ling.ucsd.edu/~barker/`

Type raising, continuations, and classical logic

Philippe de Groote
Inria-Lorraine

Abstract. There is a striking analogy between type raising, as introduced by Montague (1973), and the notion of continuation that has been developped in programming language theory in order to give compositional semantics to control operators (Stratchey and Wadsworth, 1974). In fact, this analogy is such that it is possible to see Montague's semantics as a continuation based semantics.

On the other hand, the notion of continuation allows classical logic to be given a Curry-Howard interpretation (Griffin 1990). In particular, the double negation law $((A \to \bot) \to \bot) \to A$ is provided with a computational content, which may be used to give a type logical interpretation of type lowering.

Putting the pieces of the picture together, it is possible to use "classical extensions" of the λ-calculus in order to express the semantic components of the lexical entries of Morrill's (1994) type logical grammars. This solution offers the advantage of not burdening the syntax by enforcing type raising to the worst case.

1 Type raising and continuations

Montague (1973) introduced *type raising* as a way of providing a compositional semantics to constructs that may give rise to scope ambiguities. Such constructs (typically, quantifiers) have semantic scopes that may be wider than their apparent syntactic scopes. Around the same time, computer scientists were trying to provide a compositional semantics to full jumps (i.e., 'goto' statements), which led to the discovery of *continuations* (Stratchey and Wadsworth, 1974).

Both problems are similar, and both solutions present striking similitudes. Montague's type raising is based on Leibniz's principle, which consists of identifying an entity with the set of its properties. Consequently, the type of entities e is replaced by $(e \to t) \to t$, where t is the type of propositions. In programming language theory, a *continuation semantics* (as opposed to a *direct semantics*) consists in providing the semantic function with the continuation of the program as an explicit parameter. Let P be a program, let $[\![-]\!]$ be the semantic function, and let s be some initial state. If we consider programs as state transformers, a direct semantics is such that $[\![P]\!]\, s \in \mathbf{State}$. On the other hand, a continuation semantics gives $[\![P]\!]\, s \in (\mathbf{State} \to \mathbf{State}) \to \mathbf{State}$. In fact, in both cases (type raising and continuation semantics), a type A is replaced by a type $(A \to O) \to O$, where O is the type of observable entities or facts.

2 Negative translations and classical logic

In the realm of the λ-calculus, the notion of continuation gave rise to the so-called CPS-transformations (Plotkin 1975). These are continuation-based syntactic transformations of the λ-terms that allow given evaluation strategies (typically, call-by-name or call-by-value) to be simulated.

For instance, Plotkin's call-by-value CPS-transformation is as follows:

$$\begin{aligned}
\overline{c} &= \lambda k.\, k\, c;\\
\overline{x} &= \lambda k.\, k\, x;\\
\overline{\lambda x.\, M} &= \lambda k.\, k\, (\lambda x.\, \overline{M});\\
\overline{M\, N} &= \lambda k.\, \overline{M}\, (\lambda m.\, \overline{N}\, (\lambda n.\, m\, n\, k))
\end{aligned}$$

Now, compare the following naive type logical grammar, where the lexical items are assigned a direct interpretation:

John	–	$\textsc{j}$	:	NP
Mary	–	$\textsc{m}$	:	NP
loves	–	$\lambda x.\, \lambda y.\, \textsc{love}\, y\, x$	:	$(NP \setminus S) \,/\, NP$

together with the grammar, where the lexical items are assigned a Montague-like interpretation:

John	–	$\lambda k.\, k\, \textsc{j}$	:	NP
Mary	–	$\lambda k.\, k\, \textsc{m}$	:	NP
loves	–	$\lambda f.\, \lambda g.\, f\, (\lambda x.\, g\, (\lambda y.\, \textsc{love}\, y\, x))$	:	$(NP \setminus S) \,/\, NP$

Again, the analogy between continuation and type raising is striking. The Montague-like interpretation may almost be seen as the call-by-value CPS-transform of the direct interpretation. This opens a new line of research that has been advocated in Barker's recent work (2000, 2001).

When applying a CPS-transformation to a typed λ-term, it induced another transformation at the type level (Meyer and Wand, 1985). For instance, the above CPS-transformation induces the following type transformation:

$\overline{\alpha} = (\alpha^* \to \bot) \to \bot$, where:

$$\begin{aligned}
a^* &= a, \quad \text{for } a \text{ atomic;}\\
(\alpha \to \beta)^* &= \alpha^* \to \overline{\beta}.
\end{aligned}$$

Griffin (1990) observed that these type transformations amount to double negative translations of classical logic into minimalist logic, and that it allows classical logic to be provided with a formulae-as-type interpretation. In this setting, the double negation law $((A \to \bot) \to \bot) \to A$, which corresponds to type lowering, is given a computational content by considering the absurd type $\bot$ to be the type of observable entities (this is radically different from the usual interpretation of $\bot$ as the empty type).

3 The $\lambda\mu$-calculus

Griffin's discovery gave rise to several extensions of the λ-calculus, which aim at adapting the Curry-Howard isomorphism to the case of classical logic. The $\lambda\mu$-calculus (Parigot 1992) is such a system.

The $\lambda\mu$-calculus is a strict extension of the λ-calculus. Its syntax is provided with a second alphabet of variables ($\alpha, \beta, \gamma, \ldots$ — called the μ-variables), and two additional constructs: μ-abatraction ($\mu\alpha.\, t$), and naming ($\alpha\, t$).
These constructs obey the following typing rules:

$$\frac{\alpha : \neg A \qquad t : A}{\alpha\, t \,:\, \bot} \qquad\qquad \frac{\begin{array}{c}[\alpha \,:\, \neg A]\\ t \,:\, \bot\end{array}}{\mu\alpha.\, t \,:\, A}$$

Besides β-reduction:

$(\beta)\;\; (\lambda x.\, t)\, t \;\longrightarrow\; t[x := u]$

a notion of μ-reduction is defined:

$(\mu)\;\; (\mu\alpha.\, u)\, v \;\longrightarrow\; \mu\beta.\, u[\alpha\, t_i := \beta\,(t_i\, v)]$

where $u[\alpha\, t_i := \beta\,(t_i\, v)]$ stands for the term u where each subterm of the form $\alpha\, t_i$ has been replaced by $\beta\,(t_i\, v)$. It corresponds to the following proof-theoretic reduction:

$$\dfrac{\dfrac{\begin{array}{c}\dfrac{\dfrac{}{\alpha : \neg(A \to B)}\,1 \qquad \begin{array}{c}\vdots\\ t_i : A \to B\end{array}}{\alpha\, t_i : \bot}\\ \vdots \\ u : \bot\end{array}}{\mu\alpha.\, u : A \to B}\,1 \qquad \begin{array}{c}\vdots\\ v : A\end{array}}{(\mu\alpha.\, u)\, v : B} \quad\longrightarrow\quad \dfrac{\begin{array}{c}\dfrac{\dfrac{}{\beta : \neg B}\,1 \qquad \dfrac{\begin{array}{c}\vdots\\ t_i : A\to B\end{array} \quad \begin{array}{c}\vdots\\ v : A\end{array}}{t_i\, v : B}}{\beta\,(t_i\, v) : \bot}\\ \vdots\\ u[\alpha\, t_i := \beta\,(t_i\, v)] : \bot\end{array}}{\mu\beta.\, u[\alpha\, t_i := \beta\,(t_i\, v)] : B}\,1$$

As well-known, classical logic is not naturally confluent. Consequently, there exist variants of the $\lambda\mu$-calculus that do not satisfy the Church-Rosser property (Parigot 2000). This is the case if we also consider the symmetric of the μ-reduction rule:

$(\mu')\;\; v\,(\mu\alpha.\, u) \;\longrightarrow\; \mu\beta.\, u[\alpha\, t_i := \beta\,(v\, t_i)]$

Finally, for the purpose of the example given in the next section, we also add the following simplification rules:

$(\sigma)\;\; \mu\alpha.\, u \;\longrightarrow\; u[\alpha\, t_i := t_i]$

which may be applied only to terms of type $\bot$.

4 Semantic recipes as $\lambda\mu$-terms

Dealing with a calculus that do not satisfy the Church-Rosser property is not a defect in the case of natural language semantics. Indeed, the fact that a same term may have several different normal forms allows one to deal with semantic ambiguities.

If we consider the sentential category S (or, semantically, Montague's type t) to be our domain of observable facts, the following typing judgement is derivable:

$$\dfrac{\dfrac{\dfrac{\dfrac{\textsc{Person} : e\to t \qquad \dfrac{}{x : e}\,0}{(\textsc{Person}\,x) : t} \qquad \dfrac{\dfrac{}{\alpha : e\to t}\,1 \quad \dfrac{}{x : e}\,0}{(\alpha\,x) : t}}{(\textsc{Person}\,x) \supset (\alpha\,x) : t}}{\forall x.(\textsc{Person}\,x) \supset (\alpha\,x) : t}\,0}{\mu\alpha.\, \forall x.(\textsc{Person}\,x) \supset (\alpha\,x) : e}\,1$$

This allows the following type logical lexical entries to be defined:

$$\begin{array}{llll}
\textbf{everybody} & - & \mu\alpha.\,\forall x.(\textsc{person}\,x) \supset (\alpha\,x) & : \; \mathit{NP} \\
\textbf{somebody} & - & \mu\alpha.\,\exists x.(\textsc{person}\,x) \wedge (\alpha\,x) & : \; \mathit{NP} \\
\textbf{loves} & - & \lambda x.\,\lambda y.\,\textsc{love}\,y\,x & : \; (\mathit{NP} \setminus S) \,/\, \mathit{NP}
\end{array}$$

Then, the sentence

everybody loves somebody

has only one parsing, to which is associated the following semantic reading:

$$(\lambda x.\,\lambda y.\,\textsc{love}\,y\,x)\,(\mu\alpha.\,\exists x.(\textsc{person}\,x) \wedge (\alpha\,x))\,(\mu\alpha.\,\forall x.(\textsc{person}\,x) \supset (\alpha\,x)).$$

This $\lambda\mu$-term may be considered as an underspecified representation. Indeed, its possible reductions yield two different normal forms:

$$\begin{array}{lll}
\multicolumn{2}{l}{(\lambda x.\,\lambda y.\,\textsc{love}\,y\,x)\,(\mu\alpha.\,\exists x.(\textsc{person}\,x) \wedge (\alpha\,x))\,(\mu\alpha.\,\forall x.(\textsc{person}\,x) \supset (\alpha\,x))} & \\
\longrightarrow & (\lambda y.\,\textsc{love}\,y\,(\mu\alpha.\,\exists x.(\textsc{person}\,x) \wedge (\alpha\,x)))\,(\mu\alpha.\,\forall x.(\textsc{person}\,x) \supset (\alpha\,x)) & (\beta) \\
\longrightarrow & \textsc{love}\,(\mu\alpha.\,\forall x.(\textsc{person}\,x) \supset (\alpha\,x))\,(\mu\alpha.\,\exists x.(\textsc{person}\,x) \wedge (\alpha\,x)) & (\beta) \\
\longrightarrow & (\mu\beta.\,\forall x.(\textsc{person}\,x) \supset (\beta\,(\textsc{love}\,x)))\,(\mu\alpha.\,\exists x.(\textsc{person}\,x) \wedge (\alpha\,x)) & (\mu') \\
\longrightarrow & \mu\beta.\,\forall x.(\textsc{person}\,x) \supset (\beta\,(\textsc{love}\,x\,(\mu\alpha.\,\exists y.(\textsc{person}\,y) \wedge (\alpha\,y)))) & (\mu) \\
\longrightarrow & \forall x.(\textsc{person}\,x) \supset (\textsc{love}\,x\,(\mu\alpha.\,\exists y.(\textsc{person}\,y) \wedge (\alpha\,y))) & (\sigma) \\
\longrightarrow & \forall x.(\textsc{person}\,x) \supset (\mu\alpha.\,\exists y.(\textsc{person}\,y) \wedge (\alpha\,(\textsc{love}\,x\,y))) & (\mu') \\
\longrightarrow & \forall x.(\textsc{person}\,x) \supset (\exists y.(\textsc{person}\,y) \wedge (\textsc{love}\,x\,y)) & (\sigma)
\end{array}$$

$$\begin{array}{lll}
\multicolumn{2}{l}{(\lambda x.\,\lambda y.\,\textsc{love}\,y\,x)\,(\mu\alpha.\,\exists x.(\textsc{person}\,x) \wedge (\alpha\,x))\,(\mu\alpha.\,\forall x.(\textsc{person}\,x) \supset (\alpha\,x))} & \\
\longrightarrow & (\lambda y.\,\textsc{love}\,y\,(\mu\alpha.\,\exists x.(\textsc{person}\,x) \wedge (\alpha\,x)))\,(\mu\alpha.\,\forall x.(\textsc{person}\,x) \supset (\alpha\,x)) & (\beta) \\
\longrightarrow & \textsc{love}\,(\mu\alpha.\,\forall x.(\textsc{person}\,x) \supset (\alpha\,x))\,(\mu\alpha.\,\exists x.(\textsc{person}\,x) \wedge (\alpha\,x)) & (\beta) \\
\longrightarrow & (\mu\beta.\,\forall x.(\textsc{person}\,x) \supset (\beta\,(\textsc{love}\,x)))\,(\mu\alpha.\,\exists x.(\textsc{person}\,x) \wedge (\alpha\,x)) & (\mu') \\
\longrightarrow & \mu\alpha.\,\exists y.(\textsc{person}\,y) \wedge (\alpha\,((\mu\beta.\,\forall x.(\textsc{person}\,x) \supset (\beta\,(\textsc{love}\,x)))\,y)) & (\mu') \\
\longrightarrow & \exists y.(\textsc{person}\,y) \wedge ((\mu\beta.\,\forall x.(\textsc{person}\,x) \supset (\beta\,(\textsc{love}\,x)))\,y) & (\sigma) \\
\longrightarrow & \exists y.(\textsc{person}\,y) \wedge (\mu\beta.\,\forall x.(\textsc{person}\,x) \supset (\beta\,(\textsc{love}\,x\,y))) & (\mu) \\
\longrightarrow & \exists y.(\textsc{person}\,y) \wedge (\forall x.(\textsc{person}\,x) \supset (\textsc{love}\,x\,y)) & (\sigma)
\end{array}$$

These correspond to subject and object wide scope readings, respectively.

5 conclusions

We have argued that Montague's type raising is a particular case of continuation. Consequently, continuation based formalisms, which have been developped in the context of programming language theory, may be used to deal with the sort of semantic ambiguities for which Montague invented type raising. Parigot's $\lambda\mu$-calculus is such a formalism, and we have shown how it may be used to cope with quantifier scope ambiguities. We claim that the $\lambda\mu$-calculus is particularly suitable for expressing compositional semantics of natural languages. For instance, it allows Cooper's (1983) storage to be given a type logical foundation. In fact, it allows a lot of dynamic constructs to be defined, which is of particular interest for discourse representation.

references

Barker, C. (2000) *Continuations and the nature of quantification* working paper, submitted for publication.

Barker, C. (2001) Introducing Continuations. In R. Hastings, B. Jackson, and Z. Zvolenszky, editors, *Proceedings of SALT 11*, CLC Publications, Ithaca, New York.

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

Griffin, T. G. (1990). A formulae-as-types notion of control. In *Conference record of the seventeenth annual ACM symposium on Principles of Programming Languages*, pages 47–58.

Meyer, A. and Wand, M. (1985). Continuation semantics in typed lambda-calculi (summary). In R. Parikh, editor, *Logics of Programs*, volume 193 of *Lecture Notes in Computer Science*, pages 219–224. Spinger Verlag.

Montague, R. (1973). The proper treatment of quantification in ordinary english. In J. Hintikka, J. Moravcsik, and P. Suppes, editors, *Approaches to natural language: proceedings of the 1970 Stanford workshop on Grammar and Semantics*, Dordrecht, Reidel. Reprinted: Montague, (1974, pages 247–270).

Montague, R. (1974). *Formal Philosophy: selected papers of Richard Montague, edited and with an introduction by Richmond Thomason.* Yale University Press.

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

Parigot, M. (1992). $\lambda\mu$-Calculus: an algorithmic interpretation of classical natural deduction. In A. Voronkov, editor, *Proceedings of the International Conference on Logic Programming and Automated Reasoning*, volume 624 of *Lecture Notes in Artificial Intelligence*, pages 190–201. Springer Verlag.

Parigot, M. (2000). On the computational interpretation of negation. In P.G. Clote and H. Schwichtenberg, editors, *Computer Science Logic*, volume 1862 of *Lecture Notes in Computer Science*, Springer Verlag.

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

Stratchey, C. and Wadsworth, C. (1974). Continuations a mathematical semantics for handling full jumps. Technical Report PRG-11, Oxford University, Computing Laboratory.

TYPES AS GRAPHS: CONTINUATIONS IN TYPE LOGICAL GRAMMAR

CHRIS BARKER AND CHUNG-CHIEH SHAN

ABSTRACT. Applying the programming-language concept of *continuations*, we propose a new multimodal analysis of quantification in Type Logical Grammar. Our approach naturally gives rise to a new, geometric (graph-theoretic) interpretation for in-situ quantification. The proposal also motivates the limited use of empty antecedents in derivations. In addition, because continuations are the tool of choice for reasoning about such things as evaluation order or parameter passing, our system provides a principled way to express generalizations concerning semantic side-effects within an ordinary multimodal type-logical framework. We illustrate the utility of these techniques by providing improved accounts of quantificational binding, weak crossover, *wh*-questions, superiority, and polarity licensing.

1. INTRODUCTION

In recent research, the programming-language concept of *continuations* has provided much insight into a variety of natural-language phenomena: quantification and coordination (Barker 2002a; de Groote 2001), binding and interrogation (Shan 2002; Shan and Barker 2003), focus and hypallage (Barker 2004), and polarity sensitivity (Shan 2004). As we explain in detail below, continuations are a compositional mechanism that provides expressions with access to (a portion of) their semantic context.

Most of the linguistically-oriented research involving continuations (including much of our own work apart from this paper) is couched in combinatory categorial grammars in which the main work is accomplished by type-shift operators in the style of, for example, Jacobson (1999) and Steedman (2000). In an ongoing effort to explore the role of continuations in linguistic formalisms other than combinatory categorial grammars, we propose in this paper a continuation-based approach to quantification in Type Logical Grammar. We show how continuations lead to an unusually parsimonious multimodal account of quantifier scope. In particular, we use continuations to reduce the ternary type constructor q proposed by Moortgat for in-situ quantification to multimodal type-logical grammar. The basic analysis involves one new mode, related to the default mode by two structural postulates. As we explain in §3, these postulates can be understood geometrically in terms of unitrivalent graphs.

Continuations have traditionally been used to explore such computational issues as evaluation order (such as left-to-right versus right-to-left), parameter passing (such as call-by-value versus call-by-name), and side effects in general. At first blush, logical proof is independent of evaluation order and parameter passing: a TLG derivation has no obvious notion corresponding to 'before' versus 'after', or 'value' versus 'name'. Yet, by the

This is a rough draft of June 18, 2004 (revision 106). It is not for general distribution. We would not have undertaken this project without the encouragement of David Dowty. Nor would we have gotten anywhere without many patient explanations from Philippe de Groote, the topological insights of Dylan Thurston, and Richard Moot's theorem prover for multimodal TLG analysis, made accessible through Dick Oehrle's web interface. We also gratefully acknowledge comments and suggestions from David Dowty, Gerhard Jäger, and Richard Moot. The second author is supported by the United States National Science Foundation Grant BCS-0236592.

Curry-Howard isomorphism, logical proof is the same thing as computation, so whenever evaluation order and parameter passing play a role in computation, they must play a role in logical proof. If so, we need a way to reason about evaluation order and parameter passing in TLG derivations, which are independent of these issues. This is precisely what continuations provide: a principled tool for reasoning about order of evaluation in an order-independent way, and about parameter passing in a parameter-passing-independent way.

Parigot's $\lambda\mu$-calculus (1992) is one way to enrich a logic with continuations. De Groote (2001) has applied it to quantification in natural language, in a similar spirit as this paper. But the $\lambda\mu$-calculus is fully ambidextrous, allowing both left-to-right and right-to-left evaluation; we provide here more fine-grained control over order of evaluation in a TLG context.

Moortgat (1996a) and others emphasize that TLG provides a perspicuous framework for exploring the resource-sensitivity of natural languages. Order of evaluation is just one more kind of structural resource. It is an empirical question whether natural languages are sensitive to evaluation order (and other constraints on side effects). We believe they are, and will describe below in §6 and §7 some empirical phenomena in support of this claim. Thus adding continuations to TLG enables us to explore a new realm of the Curry-Howard correspondence, and provides for the grammar-writer a new and (we argue) useful type of resource sensitivity.

In §4, we detail how we implement in-situ quantification by residuating on contexts and subexpressions. In §5, we discuss how our approach motivates the selective use of empty antecedents. In §6, we show how to fine-tune the notion of contexts for a variety of linguistic purposes.

There are several previous proposals (which we survey in §2) for reducing the general quantificational type constructor q to binary and unary connectives. Our offering has both theoretical and empirical advantages. On a theoretical level, we relate quantification in TLG to concepts from programming-language semantics, like evaluation order (§6.3) and parameter passing (§7.2). On an empirical level, we provide new TLG analyses of quantificational binding, weak crossover, *wh*-questions, superiority, and polarity licensing that improve on the predictions of previous TLG accounts (§7).

2. Quantification in Type Logical Grammar

There are a variety of approaches to quantification in TLG, including Moortgat's type constructor q (1988; 1995; 1996b) and at least three multimodal implementations of q (Moortgat 1995, 2000; Morrill 1994), briefly summarized by Bernardi (2003). The multimodal analyses all reconstruct the q operator in terms of modes and structural postulates, and our analysis below does the same. One important difference is that our analysis deliberately exploits continuations, which, as we will see, enables the grammar-writer to explicitly reason about order of evaluation and other side-effects.

The q operator constructs types of the form $q(A, B, C)$. Conceptually, this type behaves locally as if it were an expression of type A, taking scope over an expression of type B, and yielding a result expression of type C. For instance, if a quantificational NP such as *everyone* has type $q(np, s, s)$, then it functions locally as if it were a normal NP but takes scope over a clause. The type of a clause at which such an NP takes scope is again a clause. For an example of a case in which it might be appropriate to choose B distinct from C, consider a language with in-situ *wh*. If the *wh*-word is of type $q(np, s, wh)$, then it would function locally as an NP, take scope at the level of a clause, and turn the clause over which it takes scope into a question type *wh*. Moortgat (2000) gives other examples motivating analyses for which $B \neq C$.

Because q was designed to capture the behavior of quantificational elements, it naturally resembles how quantificational elements are treated in a continuation-based grammar such as Barker's (2002a) and Shan and Barker's (2003). For instance, in Shan and Barker's system (2003), quantificational NPs have type $(np \sim s) \to s$; in general, an expression of type $(A \sim B) \to C$ is something that functions locally as an A, takes scope over a constituent of type B, and turns it into an expression of type C.

The multimodal implementations of the q operator mentioned above seek to characterize how quantification depends on structural resources such as associativity or commutativity. For instance, Moortgat (1996a, page 125) argues that since quantifiers can take scope both over material to their left and to their right, the plain Lambek calculus, whether associative or not, simply does not have the expressive power to deal with quantification in a general way. He concludes that some degree of commutativity is essential, and our analysis below is consistent with that claim.

Though we will not present in detail the assumptions and mechanisms of the other multimodal accounts of quantification we are aware of, we can describe their general features.

Morrill (1994) uses three binary modes: the non-associative default mode $\circ$, an associative mode $\circ_a$, and a wrapping mode $\circ_w$. Three postulates allow the modes to interact in such a way that a quantificational NP of type $(s/_w np)\backslash_w s$ have the desired behavior.

In the first of his two analyses, Moortgat (1995) uses two binary modes and three unary modes. There are three postulates, and the type of a quantificational NP is

$$\Diamond(s/_w(\Box^{\downarrow} np\backslash_w s)). \tag{1}$$

Moortgat's second analysis (2000) has three binary modes and four single-sided postulates. The type of a quantificational NP is

$$(s/_+(np\backslash_- s)) \circ_+ (np\backslash_- np). \tag{2}$$

Note that $np\backslash_- np$, the right-hand argument of the fusion connective, in effect introduces an empty antecedent into the derivation. Empty antecedents also play an important role in our analysis of quantification, as discussed below in §5.

Our analysis has two binary modes, the default $\circ$ and an additional mode $\circledcirc$. Unlike most TLG analyses of quantification, neither mode needs to be associative for our purposes (nor do we require either mode to be non-associative). As explained below, the two modes are governed by two postulates, and the lexical type of a quantificational NP is $s/\!\!/(np\backslash\!\!\backslash s)$. Here we write $/\!\!/$ and $\backslash\!\!\backslash$ for residuation in the additional mode $\circledcirc$.

3. Fusion types as trees

We now introduce our graphical interpretation of our multimodal implementation of q. We will make the fairly abstract discussion in this section more linguistically concrete in §4.

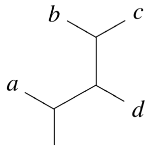
On one level, the interpretation of our system in geometric terms serves to explain the mechanics of our multimodal implementation of q. On another level, the natural geometrical interpretation that the TLG analysis lends itself to provides new insight into the logic of continuation.

Every TLG type built from atomic symbols using only $\circ$ can be uniquely drawn as a binary tree. A binary tree is an acyclic unitrivalent graph in which every leaf node but one (the *root*) is labeled with an atomic symbol. A unitrivalent graph is a graph in which every node either is a leaf or has exactly three edges. For example, the type

(3) $$A = a \circ ((b \circ c) \circ d)$$

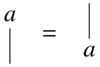
can be drawn as follows.

(4) $$a \circ ((b \circ c) \circ d) \iff$$



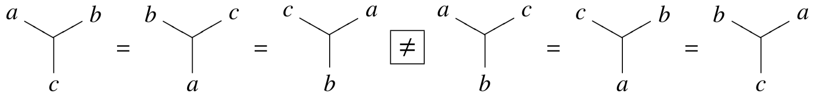
In our graphs, edges are unoriented, so the following two graphs are identical:

(5)



However, we do orient nodes by designating, for each trivalent node, one of the two cyclic orderings of its edges as clockwise. Hence each of the following graphs are equal to two others, but not to their mirror images.

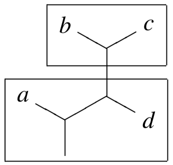
(6)



Together, these two conventions ensure that each type corresponds to only one graph.

Suppose now that A and B are two types, both built using only $\circ$, such that B appears as part of A. For instance, A might be the type $a \circ ((b \circ c) \circ d)$ as above, and B might be the type $b \circ c$. We write $A = \mathcal{K}[B]$, where $\mathcal{K}$ is the context of B relative to A; that is, $\mathcal{K} = a \circ ([\,] \circ d)$. Graphically speaking, we have decomposed A into two parts, B and $\mathcal{K}$, by ripping apart the graph at the upper vertical edge, calling the side that contains the root node $\mathcal{K}$, and calling the other side B.

(7)

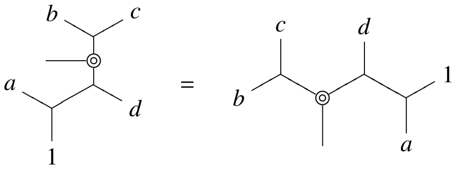


$$\iff b \circ c = B$$

$$\iff a \circ ([\,] \circ d) = \mathcal{K}$$

To represent such a decomposition as a graph, we insert a new, special node ⊚ in the middle of the edge that connects the two components.

(8)



This new node ⊚ connects the two components of the decomposition to a new root node. The old root node becomes a leaf with the special label 1. We know which side of the ⊚ corresponds to the context, since that's the side that contains the 1 node marking the position of the old root. Starting at the ⊚ node, we always put the context side right after the subexpression side (where "after" means moving clockwise).

To convert the diagram (8) back to a type, we introduce a new binary mode, the *continuation* or *context* mode. We write fusion, left residuation, and right residuation for the continuation mode as ⊚, ⑊, and ⫽. The graph in (8) thus corresponds to the formula

(9) $$(b \circ c) \circledcirc (d \circ (1 \circ a)).$$

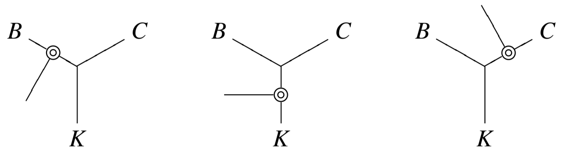
The continuation mode ⊚ treats the decomposition $A = \mathcal{K}[B]$ as a structure in its own right: $B \circledcirc K$ decomposes A into the subexpression B and the context $\mathcal{K}$.[1] In the trivial case, the graph is ripped apart at the root edge into two sides: the null context ('[ ]'), and the expression A viewed as its own (improper) subexpression. Because the null context is represented by the type 1, we henceforth treat 1 as the right identity for ⊚. Thus $B \circledcirc 1$ is logically equivalent to just B. (The presence of a right identity affects the nature of the ⊚ mode, and plays an important role in the discussion below; see especially §5.)

In the original type $A = a \circ ((b \circ c) \circ d)$ in (3), the type d follows a and fuses with $b \circ c$, whereas in (9), the type d precedes a and fuses with $1 \circ a$. It appears that our types represent a context by scrambling the original elements—in fact, by turning the original expression inside-out. But this reordering only occurs in the symbolic representation as types, as in (9). When viewed graphically, as in (8), there is no scrambling, and no turning inside-out: we have simply inserted a continuation node into the chosen edge, without reordering anything. Put another way, we have not changed the structure of the formula; we have merely changed our perspective on the formula by placing one subformula in the foreground and backgrounding the rest. Thus the context $d \circ (1 \circ a)$ is not really "inside out"—that's just what the context looks like from the perspective of the continuation node. The usefulness of the graphical interpretation is precisely that it makes the mapping from contexts to types as simple as choosing an edge.

This technique—representing a meta-level notion of context such as $\mathcal{K} = a \circ ([\,] \circ d)$ as a type-level expression like $K = d \circ (1 \circ a)$—embodies the crucial insight of continuations. We shall see that allowing the logic to operate on types representing contexts in effect provides expressions with access to their own context.

We are now ready to introduce some structural rules to relate the various equivalent ways in which the same graph can be decomposed. If we find some type of the form $B \circ C$ inside a context $\mathcal{K}$, we can consider moving either B or C into the context part of the decomposition. In other words, if $A = \mathcal{K}[B \circ C]$, then three decompositions of A are available: one that isolates B, one that isolates $B \circ C$, and one that isolates C. If the type K represents the context of $B \circ C$ (so that it contains 1 in place of the root of A), then we can depict these three decompositions as follows.

(10)



$B \circledcirc (C \circ K)$ $(B \circ C) \circledcirc K$ $C \circledcirc (K \circ B)$

[1] We use script $\mathcal{K}$ to stand for the usual notion of a context as an expression containing a hole (where 'usual' means as in discussions of programming languages, like Barendregt's presentation of the λ-calculus (1981)); and we use plain K to stand for the TLG type that we interpret as representing a context.

Since these are three decompositions of the same graph, we want them to be logically equivalent, that is, mutually derivable. To this end, we add two structural postulates.[2]

(11) $B \circledcirc (C \circ K) \dashv\vdash (B \circ C) \circledcirc K$ (Left) $\quad (B \circ C) \circledcirc K \dashv\vdash C \circledcirc (K \circ B)$ (Right)

Using these structural rules, any decomposition of a (connected) graph A can be derived from the trivial decomposition $A \circledcirc 1$, which represents A itself inside the null context (which, we repeat, is logically equivalent to just A, given that 1 is the right identity for $\circledcirc$). For example, the decomposition in (8) can be derived from the trivial decomposition in two steps.

(12) $$(a \circ ((b \circ c) \circ d)) \circledcirc 1 \overset{\text{Right}}{\dashv\vdash} ((b \circ c) \circ d) \circledcirc (1 \circ a) \overset{\text{Left}}{\dashv\vdash} (b \circ c) \circledcirc (d \circ (1 \circ a))$$

The continuation mode $\circledcirc$ and the structural postulates Left and Right expand the machinery of residuation built-in to categorial grammar beyond describing syntactic elements that take arguments leftward or rightward. We can now describe syntactic elements that take arguments not leftward or rightward, but 'inward' or 'outward', in senses to be explained in the next section. We will argue, following Barker (2002a), that this is the very essence of in-situ quantification.

4. Scope-taking as residuation on contexts and subexpressions

Let us apply the techniques developed above to some syntactic categories and lexical items of linguistic relevance. We assume a type of noun phrases (more precisely, proper nouns) np and a type of clauses s. For example, *Alice saw Bob* is a clause.

(13) $$\dfrac{\text{Alice} \vdash np \qquad \dfrac{\text{saw} \vdash (np\backslash s)/np \qquad \text{Bob} \vdash np}{\text{saw} \circ \text{Bob} \vdash np\backslash s}\,/\text{E}}{\text{Alice} \circ (\text{saw} \circ \text{Bob}) \vdash s}\,\backslash\text{E}$$

[2] Unfortunately, in the presence of these two structural postulates and the right identity 1 for the $\circledcirc$ mode, the default mode $\circ$ becomes commutative.

$$\begin{array}{l} B \circ C \overset{\text{Pop/Push}}{\dashv\vdash} (B \circ C) \circledcirc 1 \overset{\text{Left}}{\dashv\vdash} B \circledcirc (C \circ 1) \overset{\text{Pop/Push}}{\dashv\vdash} B \circledcirc ((C \circ 1) \circledcirc 1) \overset{\text{Left}}{\dashv\vdash} B \circledcirc (C \circledcirc (1 \circ 1)) \\ \hfill \| \\ C \circ B \overset{\text{Pop/Push}}{\dashv\vdash} (C \circ B) \circledcirc 1 \overset{\text{Right}}{\dashv\vdash} B \circledcirc (1 \circ C) \overset{\text{Pop/Push}}{\dashv\vdash} B \circledcirc ((1 \circ C) \circledcirc 1) \overset{\text{Right}}{\dashv\vdash} B \circledcirc (C \circledcirc (1 \circ 1)) \end{array}$$

There are at least three ways to prevent the default mode $\circ$ from commuting thus.

(1) In §5 below, we show how to translate our grammar into one that does not use the right identity 1 for the $\circledcirc$ mode, in case one would like to avoid 1 on computational or other grounds. Our translation is incomplete in a helpful way: although it encodes every other derivation in this paper, it does not encode formulas with multiple 1's fused together, like $1 \circ 1$ above. Thus $\circ$ no longer commutes in the target of the translation.

(2) Another way to avoid 1 is to remove 1 from the grammar and replace every type A throughout the lexicon that is either a top-level formula or on the numerator side of a slash with the type $A /\!\!/ (A \mathbin{\backslash\!\!\backslash} A)$. This move prevents 1 from being freely introduced by Push, as in the derivation above.

(3) We can change the Left and Right rules to

$$B \circledcirc (C \triangleleft K) \dashv\vdash (B \circ C) \circledcirc K \quad \text{(Left)} \qquad (B \circ C) \circledcirc K \dashv\vdash C \circledcirc (K \triangleright B) \quad \text{(Right)},$$

where $\triangleleft$ and $\triangleright$ are two new binary modes. Informally speaking, we distinguish between "left contexts" ($\triangleleft$) and "right contexts" ($\triangleright$) to rule out the adjacent application of Left and Right above.

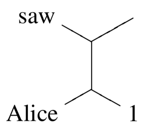
On one hand, solutions 1 and 2 are natural if one blames the presence of multiple 1's in the derivation above for making the default mode commutative. On the other hand, solution 3 is natural if one blames the commutativity on confusing Left with Right, or subexpression with context. Because solution 1 is deployed below, we leave the commutativity issue aside here.

In general, any two NPs with *saw* in between form a clause.

(14) $$\dfrac{\dfrac{}{np \vdash np}\,\text{Id} \qquad \dfrac{\text{saw} \vdash (np\backslash s)/np \qquad \dfrac{}{np \vdash np}\,\text{Id}}{\text{saw} \circ np \vdash np\backslash s}\,/\text{E}}{np \circ (\text{saw} \circ np) \vdash s}\,\backslash\text{E}$$

We proceed by drawing a (limited) analogy between what we call contexts here, and clauses containing a gap. The string *Alice saw* __ is a gapped clause. In other words, it is a context whose hole can be filled in with np to yield an s. Following the graphical approach explained in the previous section, we can represent the gapped clause as a type expression, or equivalently, as a unitrivalent graph.

(15) $(1 \circ \text{Alice}) \circ \text{saw} \iff$



We can prove this context to be a gapped clause using the structural rules in (11). Formally, as shown in (16) below, the type in (15) entails the type $np \mathbin{\backslash\!\!\backslash} s$. The latter type is the type of something that gives s when fused on the left with an np using $\circledcirc$; in other words, a gapped clause has the type of a context that encloses an np to give an s.

(16) $$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{Alice} \vdash np \qquad \dfrac{\text{saw} \vdash (np\backslash s)/np \qquad \dfrac{}{np \vdash np}\,\text{Id}}{\text{saw} \circ np \vdash np\backslash s}\,/\text{E}}{\text{Alice} \circ (\text{saw} \circ np) \vdash s}\,\backslash\text{E}}{(\text{Alice} \circ (\text{saw} \circ np)) \circledcirc 1 \vdash s}\,\text{Push}}{(\text{saw} \circ np) \circledcirc (1 \circ \text{Alice}) \vdash s}\,\text{Right}}{np \circledcirc ((1 \circ \text{Alice}) \circ \text{saw}) \vdash s}\,\text{Right}}{(1 \circ \text{Alice}) \circ \text{saw} \vdash np \mathbin{\backslash\!\!\backslash} s}\,\mathbin{\backslash\!\!\backslash}\text{I}$$

Push and Pop reflect the fact that 1 is a right identity for $\circledcirc$; they are roughly equivalent to introducing (Push) an empty antecedent into the derivation, and later eliminating (Pop) it. They are discussed in more detail below in §5, where we implement Push and Pop in terms of standard TLG technology based on unary modes.

This treatment of gapped clauses is insensitive to the linear location of the gap: it works uniformly regardless of whether the gap is at the left or right edge of the clause, unlike many (though by no means all) type-logical treatments of extraction and quantification.

We can now treat quantificational noun phrases such as *everyone*. As a syntactic element, *everyone* combines with a gapped clause enclosing it to give a complete clause. In terms of our context mode $\circledcirc$, *everyone* gives s when fused on the right with $np \mathbin{\backslash\!\!\backslash} s$. Thus we assign to it the type $s /\!\!/ (np \mathbin{\backslash\!\!\backslash} s)$. This type lets us prove the grammaticality of *Alice saw everyone*, as shown in (17). Moreover, the standard Curry-Howard meaning assignment mechanism in TLG automatically computes the denotation of the sentence to be everyone($\lambda x.\ \text{saw}(x)(\text{Alice})$).

(17) $$\dfrac{\dfrac{\dfrac{\dfrac{\text{everyone} \vdash s /\!\!/ (np \mathbin{\backslash\!\!\backslash} s) \qquad \dfrac{}{(1 \circ \text{Alice}) \circ \text{saw} \vdash np \mathbin{\backslash\!\!\backslash} s}\,(16)}{\text{everyone} \circledcirc ((1 \circ \text{Alice}) \circ \text{saw}) \vdash s}\,/\!\!/\text{E}}{(\text{saw} \circ \text{everyone}) \circledcirc (1 \circ \text{Alice}) \vdash s}\,\text{Right}}{(\text{Alice} \circ (\text{saw} \circ \text{everyone})) \circledcirc 1 \vdash s}\,\text{Right}}{\text{Alice} \circ (\text{saw} \circ \text{everyone}) \vdash s}\,\text{Pop}$$

Informally speaking, just as the function-type constructors / and \ for the default mode $\circ$ take arguments from the right and from the left, the function-type constructors $/\!\!/$ and $\backslash\!\!\backslash$ for the continuation mode $\circledcirc$ take arguments from outside (that is, residuate on surrounding contexts) and from inside (that is, residuate on enclosed subexpressions), respectively. More generally, we can reconstruct the ternary type constructor q for in-situ quantification in TLG: simply encode $q(A, B, C)$ as $C/\!\!/(A\backslash\!\!\backslash B)$. The type for *everyone* in (17) encodes $q(np, s, s)$, as expected.

Like any account of in-situ quantification implementing q, our system derives both linear and inverse scope for the sentence *Someone saw everyone*, as follows.

(18)

$$\dfrac{\dfrac{\dfrac{\text{someone} \vdash s/\!\!/(np\backslash\!\!\backslash s) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{everyone} \vdash s/\!\!/(np\backslash\!\!\backslash s) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{np \circ (\text{saw} \circ np) \vdash s}(14)}{(np \circ (\text{saw}\circ np)) \circledcirc 1 \vdash s}\text{Push}}{(\text{saw}\circ np)\circledcirc(1\circ np)\vdash s}\text{Right}}{np\circledcirc((1\circ np)\circ\text{saw})\vdash s}\text{Right}}{(1\circ np)\circ\text{saw}\vdash np\backslash\!\!\backslash s}\backslash\!\!\backslash\text{I}}{\text{everyone}\circledcirc((1\circ np)\circ\text{saw})\vdash s}/\!\!/\text{E}}{(\text{saw}\circ\text{everyone})\circledcirc(1\circ np)\vdash s}\text{Right}}{(np\circ(\text{saw}\circ\text{everyone}))\circledcirc 1\vdash s}\text{Right}}{np\circledcirc((\text{saw}\circ\text{everyone})\circ 1)\vdash s}\text{Left}}{(\text{saw}\circ\text{everyone})\circ 1\vdash np\backslash\!\!\backslash s}\backslash\!\!\backslash\text{I}}{\text{someone}\circledcirc((\text{saw}\circ\text{everyone})\circ 1)\vdash s}/\!\!/\text{E}}{(\text{someone}\circ(\text{saw}\circ\text{everyone}))\circledcirc 1\vdash s}\text{Left}}{\text{someone}\circ(\text{saw}\circ\text{everyone})\vdash s}\text{Pop}$$

(19)

$$\dfrac{\dfrac{\dfrac{\dfrac{\text{everyone} \vdash s/\!\!/(np\backslash\!\!\backslash s) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{someone} \vdash s/\!\!/(np\backslash\!\!\backslash s) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{np \circ (\text{saw} \circ np) \vdash s}(14)}{(np \circ (\text{saw}\circ np)) \circledcirc 1 \vdash s}\text{Push}}{np\circledcirc((\text{saw}\circ np)\circ 1)\vdash s}\text{Left}}{(\text{saw}\circ np)\circ 1\vdash np\backslash\!\!\backslash s}\backslash\!\!\backslash\text{I}}{\text{someone}\circledcirc((\text{saw}\circ np)\circ 1)\vdash s}/\!\!/\text{E}}{(\text{someone}\circ(\text{saw}\circ np))\circledcirc 1\vdash s}\text{Left}}{(\text{saw}\circ np)\circledcirc(1\circ\text{someone})\vdash s}\text{Right}}{np\circledcirc((1\circ\text{someone})\circ\text{saw})\vdash s}\text{Right}}{(1\circ\text{someone})\circ\text{saw}\vdash np\backslash\!\!\backslash s}\backslash\!\!\backslash\text{I}}{\text{everyone}\circledcirc((1\circ\text{someone})\circ\text{saw})\vdash s}/\!\!/\text{E}}{(\text{saw}\circ\text{everyone})\circledcirc(1\circ\text{someone})\vdash s}\text{Right}}{(\text{someone}\circ(\text{saw}\circ\text{everyone}))\circledcirc 1\vdash s}\text{Right}}{\text{someone}\circ(\text{saw}\circ\text{everyone})\vdash s}\text{Pop}$$

Ignoring the structural rules Push, Pop, Left, and Right, these derivations correspond exactly to ones using q. The quantifier that takes its context as argument earlier takes wider scope (reading the proofs bottom-up).

Without further stipulation, our system handles quantificational NPs in possessive position, as in *Everyone's mother saw someone's father*. This success is enjoyed by every TLG treatment of quantification we are aware of, but is by no means typical of other frameworks, such as LF approaches based on Quantifier Raising, or even systems fairly closely related to TLG such as Jacobson's (1999) combinatory categorial grammar, as discussed by Barker (2002b).

5. Empty antecedents

Throughout this paper, we assume a right identity 1 for the continuation mode $\circledcirc$, or equivalently, two structural rules Push and Pop that operate to the right of the $\circledcirc$ mode. A logic with 1 corresponds to a modal frame with *right identity* (Restall 2000) and a programming language with *delimited* continuations (Danvy and Filinski 1989; Felleisen 1988; Sitaram and Felleisen 1990). Assuming that 1 is present is equivalent to allowing the antecedent to be empty in the conclusion of the $\backslash\!\!\backslash$I rule, because we can treat each occurrence of 1 as just shorthand for the type $s\backslash\!\!\backslash s$ or some other type of the form $A\backslash\!\!\backslash A$, which can be derived using such a permissive $\backslash\!\!\backslash$I rule.

However, dating back at least to Lambek's original paper on his linguistic calculus (1958), there is a tradition of prohibiting empty antecedents in TLG. There is some linguistic justification for the prohibition. For instance, assume that *very* has a category appropriate for an adjective modifier, say, $(n/n)/(n/n)$ as in *a very tall man*. Then since the identity type n/n would be a theorem if we allow empty antecedents, we incorrectly derive the ungrammatical **a very man*.

Yet there is no logical reason why empty antecedents shouldn't be allowed (Moot 2002, page 74). Nor is the present paper the first to find empty antecedents linguistically useful in certain situations. For one, Moot (2002, page 85) suggests that allowing empty antecedents can be convenient given a certain analysis of pied piping. For another, consider the type of Moortgat's (2000) in-situ-quantificational NP, shown in (2) on page 3 and repeated below.

(2) $$(s/_{+}(np\backslash_{-}s)) \circ_{+} (np\backslash_{-}np).$$

Moortgat informally says that the role of the identity type $np\backslash_{-}np$ is to "stay in place" while the quantificational part $s/_{+}(np\backslash_{-}s)$ "travels upwards". If empty antecedents (or a right identity) were available for Moortgat's $\circ_{-}$ mode, the lexical type of a quantificational NP could be simply $s/_{+}(np\backslash_{-}s)$. Thus Moortgat implicitly motivates making empty antecedents available for the $\circ_{-}$ mode, at least under some circumstances.

The useful derivations in our system never fuse two 1's together, as in $1 \circ 1$ or more complex structures like $1 \circ ((1 \circ 1) \circ 1)$. It turns out that the presence of the right identity 1 can thus be simulated at the cost of proliferating combination modes, structural postulates, and lexical types. The basic idea of this simulation is to translate types with 1 into logically equivalent types without 1. We add two new unary modes to the grammar:

(20) $\Diamond_{\circ 1}A$ to replace $A \circ 1$, $\quad \Diamond_{1\circ}A$ to replace $1 \circ A$.

The computation in (21) then instructs us to add the two postulates in (22).

(21)

$$\begin{tikzcd}
B \circledcirc (C \circ 1) \arrow[r, dash, "\text{Left}", "\dashv\vdash"'] & (B \circ C) \circledcirc 1 \arrow[r, dash, "\text{Right}", "\dashv\vdash"'] \arrow[d, dash, "\text{Pop/Push}", "\dashv\vdash"'] & C \circledcirc (1 \circ B) \\
 & B \circ C &
\end{tikzcd}$$

(22) $B \circledcirc \Diamond_{\circ 1}C \dashv\vdash B \circ C$ (Left′) $\quad B \circ C \dashv\vdash C \circledcirc \Diamond_{1\circ}B$ (Right′)

Finally, if any lexical item takes a type of the form $A\backslash\!\!\backslash A$ or $A/\!\!/A$ as argument, then another lexical type needs to be added that does not take that argument. For example, a lexical item of type $wh/\!\!/(s\backslash\!\!\backslash s)$ must be made ambiguous between that type and the type *wh*. The semantic value of the argument missing from the latter version is the identity function on s.

To summarize: the selective use of empty antecedents (or equivalently, the presence of an identity for the $\circledcirc$ mode) is motivated by this and other linguistic analyses in TLG, but can be obviated by adding the two unary modes in (20) and the two postulates in (22), and introducing lexical polysemy.[3]

6. Refining notions of context

As explained in §3, contexts and subexpressions can be represented as types in multimodal TLG using a new mode and two structural postulates with a geometric interpretation. The postulates for this new mode $\circledcirc$—be they described pictorially as in (10), or symbolically as in (11)—specify a notion of context that allows descending recursively into either the left branch or the right branch of a $\circ$-fused constituent. That is:[4]

(23) A context is one of the following.
  a. A "hole to the left" $[\,]\circ C$, embedded in some outer context $\mathcal{K}$.
  This alternative is represented by the Left postulate as $C\circ K$, and constructs contexts of the form $((\cdots\circ C)\circ B)\circ A$.
  b. A "hole to the right" $B\circ[\,]$, embedded in some outer context $\mathcal{K}$.
  This alternative is represented by the Right postulate as $K\circ B$, and constructs contexts of the form $A\circ(B\circ(C\circ\cdots))$.
  c. The null context $[\,]$, the ground case for recursion. This alternative is represented by the Push and Pop postulates as 1.
  In (a) and (b) above, the type K represents the outer context $\mathcal{K}$.

By changing the structural postulates relating the continuation mode $\circledcirc$ to other modes, we can express different notions of context: adding postulates broadens the notion; removing postulates constrains the notion. Different notions of context can be mixed in the same grammar by using one $\circledcirc$-like mode for each notion. This section demonstrates the utility of this flexibility with three linguistic applications. We present these applications as separate amendments to the basic continuations analysis, but these amendments are fully compatible with each other.

6.1. Restrictors.

6.1. Restrictors. Having analyzed simple quantificational NPs like *everyone* and *someone* in §4, we now turn to quantifiers that take a common noun or a more complex restrictor constituent as argument, as in *most schools* or *every dean of a school*. We assume that common nouns mean functions from individuals (type np) to propositions or truth values (type s), but they cannot directly combine with NPs in the default mode (hence **Harvard school* is unacceptable). Thus we introduce a new mode $\circ_n$ (the letter n being mnemonic for "noun") and assign nouns such as *school* to the category $np\backslash_n s$.[5] The new mode $\circ_n$ is an internal one, so the direction of this slash is arbitrary, but it is chosen in analogy with the direction of the default-mode slash in the type $np\backslash s$ of an intransitive verb.

A quantifier like *every* combines with a noun to the right to form a quantificational NP. Hence we let *every* take the type

(24) $$(s/\!\!/(np\backslash\!\!\backslash s))/(np\backslash_n s).$$

It is then straightforward to derive the sentence *every school griped*.

(25)
$$\dfrac{\dfrac{\text{every}\vdash(s/\!\!/(np\backslash\!\!\backslash s))/(np\backslash_n s)\quad \text{school}\vdash np\backslash_n s}{\text{every}\circ\text{school}\vdash s/\!\!/(np\backslash\!\!\backslash s)}\,/\text{E}\qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{np\vdash np}\,\text{Id}\quad \text{griped}\vdash np\backslash s}{np\circ\text{griped}\vdash s}\,\backslash\text{E}}{(np\circ\text{griped})\circledcirc 1\vdash s}\,\text{Push}}{np\circledcirc(\text{griped}\circ 1)\vdash s}\,\text{Left}}{\text{griped}\circ 1\vdash np\backslash\!\!\backslash s}\,\backslash\!\!\backslash\text{I}}{\dfrac{\dfrac{(\text{every}\circ\text{school})\circledcirc(\text{griped}\circ 1)\vdash s}{((\text{every}\circ\text{school})\circ\text{griped})\circledcirc 1\vdash s}\,\text{Left}}{(\text{every}\circ\text{school})\circ\text{griped}\vdash s}\,\text{Pop}}\,/\!\!/\text{E}$$

If we add the lexical items

(26) $$\text{dean of}\vdash(np\backslash_n s)/np,$$

(27) $$\text{Harvard}\vdash np,$$

then essentially the same derivation generates *Every dean of Harvard griped.*

How does the $\circ_n$ mode interact with the continuation mode $\circledcirc$? We broaden our notion of context to allow descending recursively into either branch of a $\circ_n$-fused constituent. That is, we revise (23) to

(28) Unchanged cases:
  a. A "hole to the left" $[\,]\circ C$, embedded in some outer context $\mathcal{K}$.
  b. A "hole to the right" $B\circ[\,]$, embedded in some outer context $\mathcal{K}$.
  c. The null context $[\,]$, the ground case for recursion.
  Added cases:
  d. A "hole to the left" $[\,]\circ_n C$, embedded in some outer context $\mathcal{K}$.
  e. A "hole to the right" $B\circ_n[\,]$, embedded in some outer context $\mathcal{K}$.

by adding the following structural postulates alongside those in (11).

(29) $B\circledcirc(C\circ_n K)\dashv\vdash(B\circ_n C)\circledcirc K\quad(\text{Left}_n)\qquad(B\circ_n C)\circledcirc K\dashv\vdash C\circledcirc(K\circ_n B)\quad(\text{Right}_n)$

A welcome consequence of this revision is that the scope ambiguity between two quantifiers is predicted even when one is located within the other's restrictor. This ambiguity is illustrated in the following sentence.

(30) Every dean of a school griped.

[3] Because the postulates in (22) are *expanding*, the grammar thus extended is no longer guaranteed to be PSPACE-parsable by virtue of Moot's PSPACE-completeness result for $\mathrm{NL}\Diamond_{\mathcal{R}^-}$ (2002, §9.2). If non-expanding postulates are desired, empty antecedents can still be obviated, at the cost of further complicating the grammar: add yet another unary mode $\Diamond_\circ$, and replace $B\circ C$ with $\Diamond_\circ(B\circ C)$ throughout.

[4] Borrowing notation from the study of programming languages, this notion of context can be expressed more succinctly using the context-free rule $[\,]\ ::=\ [[\,]\circ A]\ \big|\ [A\circ[\,]]\ \big|\ [\,]$.

[5] We cannot reuse the continuation mode $\circledcirc$ for the $\circ_n$ mode and assign *school* to the category $np\backslash\!\!\backslash s$. If we do, then the Right_n postulate that we are about to add in (29) would predict that the strings *Alice is a dean of Harvard* and **Harvard is an Alice dean of* are equal in grammaticality and meaning:

$$\begin{array}{cl}
\text{Alice}\circledcirc(\text{dean of}\circ\text{Harvard}) & \\ \hline\hline
(\text{Alice}\circledcirc(\text{dean of}\circ\text{Harvard}))\circledcirc 1 & \text{Push} \\ \hline\hline
((\text{Alice}\circ\text{dean of})\circledcirc\text{Harvard})\circledcirc 1 & \text{Left} \\ \hline\hline
\text{Harvard}\circledcirc(1\circledcirc(\text{Alice}\circ\text{dean of})) & \text{Right}_n \\ \hline\hline
\text{Harvard}\circledcirc((1\circ\text{Alice})\circledcirc\text{dean of}) & \text{Left} \\ \hline\hline
\text{Harvard}\circledcirc(\text{Alice}\circledcirc(\text{dean of}\circ 1)) & \text{Right} \\ \hline\hline
\text{Harvard}\circledcirc((\text{Alice}\circ\text{dean of})\circledcirc 1) & \text{Left} \\ \hline\hline
\text{Harvard}\circledcirc(\text{Alice}\circ\text{dean of}) & \text{Pop}
\end{array}$$

As Dalrymple et al. note (1999, §2.5.1), it can be tricky to account for linear scope in this sentence because there is no overt clausal boundary ("syntactic unit at the f-structure level") under *every dean* where *a school* can take scope. Nevertheless, the "imaginary clausal boundary" in the category $np\backslash_n s$ for nouns suffices for *a school* to take scope over. We can prove that *dean of a school* has the type $np\backslash_n s$, just like a common noun.

(31)
$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{a} \vdash (s/\!\!/(np\backslash\!\!\backslash s))/(np\backslash_n s) \quad \text{school} \vdash np\backslash_n s}{\text{a} \circ \text{school} \vdash s/\!\!/(np\backslash\!\!\backslash s)}\,/\text{E} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{np \vdash np}\,\text{Id} \quad \dfrac{\text{dean of} \vdash (np\backslash_n s)/np \quad \dfrac{}{np \vdash np}\,\text{Id}}{\text{dean of} \circ np \vdash np\backslash_n s}\,/\text{E}}{np \circ_n (\text{dean of} \circ np) \vdash s}\,\backslash_n\text{E}}{(np \circ_n (\text{dean of} \circ np)) \circledcirc 1 \vdash s}\,\text{Push}}{(\text{dean of} \circ np) \circledcirc (1 \circ_n np) \vdash s}\,\text{Right}_n}{np \circledcirc ((1 \circ_n np) \circ \text{dean of}) \vdash s}\,\text{Right}}{(1 \circ_n np) \circ \text{dean of} \vdash np\backslash\!\!\backslash s}\,\backslash\!\!\backslash\text{I}}{(\text{a} \circ \text{school}) \circledcirc ((1 \circ_n np) \circ \text{dean of}) \vdash s}\,/\!\!/\text{E}}{(\text{dean of} \circ (\text{a} \circ \text{school})) \circledcirc (1 \circ_n np) \vdash s}\,\text{Right}}{(np \circ_n (\text{dean of} \circ (\text{a} \circ \text{school}))) \circledcirc 1 \vdash s}\,\text{Right}_n}{np \circ_n (\text{dean of} \circ (\text{a} \circ \text{school})) \vdash s}\,\text{Pop}}{\text{dean of} \circ (\text{a} \circ \text{school}) \vdash np\backslash_n s}\,\backslash_n\text{I}$$

The Right_n structural rule is crucial to this derivation. As indicated by the $/\!\!/\text{E}$ deduction above, *a school* takes scope entirely within this complex noun, so inserting this proof into a derivation for (30) generates the linear-scope reading. The inverse-scope reading is also generated, without using the additional postulates in (29). The top of that derivation proves

(32) $$(\text{every} \circ (\text{dean of} \circ np)) \circ \text{griped} \vdash s.$$

6.2. **Islands.** If we add a new mode of combination to a grammar without also adding any postulate relating the new mode to the continuation mode $\circledcirc$, then contexts would be unable to cross the new mode. In other words, the new mode would be an island.

For example, many people believe that quantificational NPs cannot take scope outside of a tensed clause. Tensed clauses can be made into scope islands in the standard TLG way using a pair of (external) unary modalities $\Diamond_i$ and $\Box_i^{\downarrow}$. For instance, if the lexical type of *thought* is $(\Box_i^{\downarrow}(np\backslash s))/s$, then *Someone thought everyone left* has only the scope reading on which *someone* takes wides scope over *everyone*. This is because

(33) $$np \circ (\text{thought} \circ (np \circ \text{left})) \vdash s$$

is not derivable; only

(34) $$np \circ \Diamond_i(\text{thought} \circ (np \circ \text{left})) \vdash s$$

is (using the derivability relation $\Diamond_i\Box_i^{\downarrow}(np\backslash s) \vdash np\backslash s$). Without a structural postulate relating $\Diamond_i$ to $\circledcirc$, *everyone* in the embedded subject position cannot take its entire surrounding context up to the matrix level as its scope argument. In other words, *thought everyone left* is a scope island. Another way to implement the generalization that tensed clauses are scope islands is to let *thought* have the lexical type $(np\backslash s)/\Diamond_i s$, so that (33) is not derivable (and nor is (34)) but

(35) $$np \circ (\text{thought} \circ \Diamond_i(np \circ \text{left})) \vdash s$$

is derivable.

6.3. **Linear order.** The linear order of quantifiers in a sentence can affect its interpretation. For example, it is frequently observed that Mandarin quantifiers tend to take surface scope (Aoun and Li 1993; Huang 1982), and quantifier scope seems to be overtly expressed by syntactic raising in Hungarian (Szabolcsi 1997). Linguistic phenomena closely related to quantification, such as binding, interrogation, and polarity sensitivity, also often behave asymetrically with respect to the location of quantifiers. The continuations view of in-situ quantification captures quantifier ordering through the following refinement of the notion of context. This refinement recapitulates the use of continuations to study *evaluation order* in programming languages (Meyer and Wand 1985, page 223; Danvy and Filinski 1989, page 15; Papaspyrou 1998); the same concept of evaluation order is applied again in §7 below to other natural language phenomena.

The inverse-scope derivation (19) goes through because *everyone* in object position is able to take as argument its context

(36) $$(1 \circ \text{someone}) \circ \text{saw},$$

or, written equivalently,

(37) $$\text{someone} \circ (\text{saw} \circ [\ ]).$$

In general, the scope of one quantifier contains another just in case the latter appears in the context argument of the former. Imagine for the moment that we are studying a language that resembles English but mandates linear scope. To rule out inverse scope, we can refine our notion of context so as to rule out any quantifier linearly located to the left of the hole. Then (37), in which the quantifier *someone* occurs to the left of the hole, would no longer be a legitimate context, whereas

(38) $$\text{Alice} \circ (\text{saw} \circ [\ ])$$

and

(39) $$[\ ] \circ (\text{saw} \circ \text{everyone})$$

would still be considered contexts. (The context (38) is used in (16) and (17) to derive *Alice saw everyone*. The context (39) is used in (14) and (18) to derive the linear-scope reading of *someone saw everyone*.)

We can implement this idea using a unary modality. Following programming-language terminology, we call an expression *pure* if it contains no quantifier, or *impure* otherwise. To distinguish pure expressions from impure ones, we tag the types of pure expressions with a unary modality $\Diamond$. Any formula can be turned pure by embedding it under $\Diamond$ using the T postulate.

(40) $$A \vdash \Diamond A \quad (\text{T})$$

The T postulate is analogous to *quotation* or *staging* in programming languages, which turns executable code into static data. Two quotations can be concatenated using the K′ postulate.

(41) $$\Diamond B \circ \Diamond C \vdash \Diamond(B \circ C) \quad (\text{K}')$$

Whereas the other rules introduced so far are double-sided, these rules are single-sided.[6]

We consider a derivation complete if it culminates in the type $\Diamond s$, not just s. The type $\Diamond s$ signifies a pure clause rather than a quantifier over clauses (propositions). By contrast,

[6] These rules and their names are explained by Moot (2002, pages 49 and 166). Curiously, the present work seems to be the first linguistic application of K′.

everyone and *someone* are impure: their type $\Diamond s /\!\!/ (np \backslash\!\!\backslash \Diamond s)$ is not surrounded by $\Diamond$, though the propositions they quantify over and finally produce are pure (hence the $\Diamond$ in $\Diamond s$).

To rule out inverse-scope contexts like (37), we modify our notion of context to require that the left branch of a $\circ$-fused constituent be pure before descending recursively into the right branch. That is, we revise (23) to

(42) Unchanged case:
a. A “hole to the left” $[\,] \circ C$, embedded in some outer context $\mathcal{K}$.
Tightened case:
b. A “hole to the right” $\Diamond B \circ [\,]$, embedded in some outer context $\mathcal{K}$.
Unchanged case:
c. The null context $[\,]$, the ground case for recursion.

In terms of structural postulates, we replace the Right postulate from (11) with a more specific instance.

(43) $B \circledcirc (C \circ K) \dashv\vdash (B \circ C) \circledcirc K$ (Left) $\qquad (\Diamond B \circ C) \circledcirc K \dashv\vdash C \circledcirc (K \circ \Diamond B)$ (Right)

With these changes to the grammar, only the linear-scope reading for *Someone saw everyone* (of type $\Diamond s$) remains derivable. The proof is shaped exactly as in (18), except the T and K′ postulates above are used after (14) to prove $\Diamond np \circ (\Diamond \text{saw} \circ np) \vdash \Diamond s$.[7]

(44)

$$
\dfrac{\text{someone} \vdash \Diamond s /\!\!/ (np \backslash\!\!\backslash \Diamond s) \qquad
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{everyone} \vdash \Diamond s /\!\!/ (np \backslash\!\!\backslash \Diamond s) \qquad
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{np \circ (\text{saw} \circ np) \vdash s}\,(14)}{\Diamond(np \circ (\text{saw} \circ np)) \vdash \Diamond s}\,\Diamond\text{I}}{\Diamond np \circ \Diamond(\text{saw} \circ np) \vdash \Diamond s}\,\text{K}'}{\Diamond np \circ (\Diamond \text{saw} \circ \Diamond np) \vdash \Diamond s}\,\text{K}'}{\Diamond np \circ (\Diamond \text{saw} \circ np) \vdash \Diamond s}\,\text{T}}{(\Diamond np \circ (\Diamond \text{saw} \circ np)) \circledcirc 1 \vdash \Diamond s}\,\text{Push}}{(\Diamond \text{saw} \circ np) \circledcirc (1 \circ \Diamond np) \vdash \Diamond s}\,\text{Right}}{np \circledcirc ((1 \circ \Diamond np) \circ \Diamond \text{saw}) \vdash \Diamond s}\,\text{Right}}{(1 \circ \Diamond np) \circ \Diamond \text{saw} \vdash np \backslash\!\!\backslash \Diamond s}\,\backslash\!\!\backslash\text{I}}{\text{everyone} \circledcirc ((1 \circ \Diamond np) \circ \Diamond \text{saw}) \vdash \Diamond s}\,/\!\!/\text{E}}{(\Diamond \text{saw} \circ \text{everyone}) \circledcirc (1 \circ \Diamond np) \vdash \Diamond s}\,\text{Right}}{(\Diamond np \circ (\Diamond \text{saw} \circ \text{everyone})) \circledcirc 1 \vdash \Diamond s}\,\text{Right}}{(np \circ (\Diamond \text{saw} \circ \text{everyone})) \circledcirc 1 \vdash \Diamond s}\,\text{T}}{(np \circ (\text{saw} \circ \text{everyone})) \circledcirc 1 \vdash \Diamond s}\,\text{T}}{np \circledcirc ((\text{saw} \circ \text{everyone}) \circledcirc 1) \vdash \Diamond s}\,\text{Left}}{(\text{saw} \circ \text{everyone}) \circledcirc 1 \vdash np \backslash\!\!\backslash \Diamond s}\,\backslash\!\!\backslash\text{I}}{\dfrac{\dfrac{\text{someone} \circledcirc ((\text{saw} \circ \text{everyone}) \circledcirc 1) \vdash \Diamond s}{(\text{someone} \circ (\text{saw} \circ \text{everyone})) \circledcirc 1 \vdash \Diamond s}\,\text{Left}}{\text{someone} \circ (\text{saw} \circ \text{everyone}) \vdash \Diamond s}\,\text{Pop}}\,/\!\!/\text{E}
$$

The inverse-scope derivation (19) is no longer valid because the Right postulate, restricted in (43), does not apply since *someone* is impure.

What we have just seen is that a linguistic preference for linear scope reflects a computational preference for left-to-right evaluation. Of course, in many languages (including

[7] See Bernardi’s thesis (2002, page 50) for the $\Diamond$I rule used in this derivation, as well as the other natural-deduction rules for the unary operators $\Diamond$ and $\Box^{\downarrow}$, namely $\Diamond$E, $\Box^{\downarrow}$I, and $\Box^{\downarrow}$E.

English), inverse scope is available. That does not mean that order-of-evaluation effects are absent—it only means that, if they are present, they are more subtle. In the rest of this section, we examine one way to reintroduce inverse scope that gives rise to favorable empirical consequences in §7 below.

The basic idea is to treat inverse scope as an instance of *multistage programming* (see Taha and Nielsen 2003 and references therein). A multistage program is a program that generates another program (and then runs it, usually). The generating program is said to execute in an *earlier* or *outer* stage, and the generated program is said to execute in a *later* or *inner* stage. (More than two stages are also possible.) Evaluation in each stage is ordered separately: later-stage code runs only after earlier-stage code finishes generating it. Informally speaking, then, we can treat the inverse-scope reading of *Someone saw everyone* as the following outer-stage program.

(45) Run the program consisting of the word *someone*, the word *saw*, and everyone.

Italics is significant here: The sentence above only uses one quantifier (*everyone*). It also mentions a word (*someone*) that is a quantifier, but does not use it. If the domain of people under discussion is Alice, Bob, and Carol, then (45) conjoins the meanings of the sentences *Someone saw Alice*, *Someone saw Bob*, and *Someone saw Carol*. These three sentences are the inner-stage programs.

A typical multistage programming language provides facilities for: creating programs (by *quotation* or *staging*); combining programs (by concatenation); and running programs (by *unquotation* or *evaluation*). The computational intuition behind unquotation is that a quoted value is an inactive piece of program text that does not execute until the “wrapping” $\Diamond$ is removed. Removing the wrapping turns inactive data into active code. We call this step “unquotation” despite the fact that “evalution” or “eval” is the commonly used term for the same concept in the staged programming literature, to avoid confusion with the concept of evaluation order just discussed.

The T and K′ postulates in (40) and (41) above are our facilities for quoting and concatenating programs, respectively. To run programs, we add the following Unquote rule.[8]

(46) $\Diamond\Diamond_u A \vdash \Diamond_u A$ (Unquote).

This rule makes sense because the type of a quoted program is enclosed in a $\Diamond$, and to run a program is to remove that $\Diamond$. Although quotation applies freely (using T), unquotation does not. Rather, unquotation is restricted to types of the form $\Diamond_u A$. Here $\Diamond_u$ is a unary modality that marks those types that are allowed as top-level types of quoted programs; this modality can be thought of as a feature checked by the Unquote rule. In particular, we hereafter take the clause type s to be shorthand for $\Diamond_u s'$, so that we can derive it from the quoted clause type $\Diamond s$ (shorthand for $\Diamond\Diamond_u s'$) using the Unquote rule. Here s' is an atomic formula distinct from s.[9]

In the presence of Unquote, the inverse-scope reading of *Someone saw everyone* is once again derivable, as follows. (Because s and $\Diamond s$ entail each other in the presence of

[8] In view of Moot’s PSPACE-completeness result for $\mathrm{NL}\Diamond_{\mathcal{R}^-}$ (2002, §9.2), the K′ and Unquote postulates present computational difficulties for practical parsing applications because they are *expanding*. As it turns out, we can avoid expanding rules using the same technique as in footnote 3 on page 10: add a unary mode $\Diamond_i$ (where i stands for “impure”), and put $\Diamond_i$ around every subformula not surrounded by $\Diamond$.

[9] In this paper, s is the only type that can be unquoted, that is, the only type of the form $\Diamond_u A$. A treatment of polarity licensing that is less simplistic than the one in §7.3 calls for multiple clause types that can be unquoted (Shan 2004).

Unquote, we can restore the lexical types of *someone* and *everyone* from $\Diamond s /\!\!/ (np \backslash\!\!\backslash \Diamond s)$ back to $s /\!\!/ (np \backslash\!\!\backslash s)$.)

(47)

$$\dfrac{\text{everyone} \vdash s/\!\!/(np\backslash\!\!\backslash s) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{someone} \vdash s/\!\!/(np\backslash\!\!\backslash s) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{np \circ (\text{saw} \circ np) \vdash s}\,(14)}{(np \circ (\text{saw} \circ np)) \circledcirc 1 \vdash s}\,\text{Push}}{np \circledcirc ((\text{saw} \circ np) \circ 1) \vdash s}\,\text{Left}}{(\text{saw} \circ np) \circ 1 \vdash np\backslash\!\!\backslash s}\,\backslash\!\!\backslash\text{I}}{\text{someone} \circledcirc ((\text{saw} \circ np) \circ 1) \vdash s}\,/\!\!/\text{E}}{(\text{someone} \circ (\text{saw} \circ np)) \circledcirc 1 \vdash s}\,\text{Left}}{\text{someone} \circ (\text{saw} \circ np) \vdash s}\,\text{Pop}}{\Diamond(\text{someone} \circ (\text{saw} \circ np)) \vdash \Diamond s}\,\Diamond\text{I} \qquad \dfrac{\dfrac{}{s \vdash s}\,\text{Id}}{\Diamond s \vdash s}\,\text{Unquote}}{\Diamond(\text{someone} \circ (\text{saw} \circ np)) \vdash s}\,\Diamond\text{E}}{\Diamond\text{someone} \circ \Diamond(\text{saw} \circ np) \vdash s}\,\text{K}'}{\Diamond\text{someone} \circ (\Diamond\text{saw} \circ \Diamond np) \vdash s}\,\text{K}'}{\Diamond\text{someone} \circ (\Diamond\text{saw} \circ np) \vdash s}\,\text{T}}{(\Diamond\text{someone} \circ (\Diamond\text{saw} \circ np)) \circledcirc 1 \vdash s}\,\text{Push}}{(\Diamond\text{saw} \circ np) \circledcirc (1 \circ \Diamond\text{someone}) \vdash s}\,\text{Right}}{np \circledcirc ((1 \circ \Diamond\text{someone}) \circ \Diamond\text{saw}) \vdash s}\,\text{Right}}{(1 \circ \Diamond\text{someone}) \circ \Diamond\text{saw} \vdash np\backslash\!\!\backslash s}\,\backslash\!\!\backslash\text{I}}{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{everyone} \circledcirc ((1 \circ \Diamond\text{someone}) \circ \Diamond\text{saw}) \vdash s}{(\Diamond\text{saw} \circ \text{everyone}) \circledcirc (1 \circ \Diamond\text{someone}) \vdash s}\,\text{Right}}{(\Diamond\text{someone} \circ (\Diamond\text{saw} \circ \text{everyone})) \circledcirc 1 \vdash s}\,\text{Right}}{\Diamond\text{someone} \circ (\Diamond\text{saw} \circ \text{everyone}) \vdash s}\,\text{Pop}}{\text{someone} \circ (\Diamond\text{saw} \circ \text{everyone}) \vdash s}\,\text{T}}{\text{someone} \circ (\text{saw} \circ \text{everyone}) \vdash s}\,\text{T}}\,/\!\!/\text{E}$$

As noted above, the Unquote rule is necessary to derive this reading if the Right rule is restricted as in (43). More precisely, in order for one quantifier to take inverse scope over another, the Unquote rule must apply to the clause produced by the narrower-scope quantifier (here *someone*).

The identity types (the 1's) and the unary modes add considerable complexity to the basic analysis of *Someone saw everyone* given above in (18). The payoff, as we shall see in the next section, is that these complications allow control over order of evaluation, which provides an account of fairly subtle and complex linguistic phenomena.

7. Beyond quantification

In this section, we survey continuation-based analyses of several linguistic phenomena, and sketch how those analyses can be transliterated into the TLG approach developed in this paper. The phenomena discussed are quite intricate, and we cannot hope to be complete or comprehensive here—many additional details are available in the papers cited. Rather, our main purpose is to show how continuations can offer explanatory insights previously unavailable in TLG, as well as improve the empirical coverage as compared to previous TLG accounts of the same phenomena.

7.1. **Binding and crossover.**

In another manuscript (Shan and Barker 2003), we argue that order of evaluation provides a unified explanation for two distinct phenomena in English, weak crossover and superiority. We will compare our approach to Jäger's (2001) TLG analysis of weak crossover. In this case, the new explanation is the claim that weak crossover and superiority stem from the same underlying mechanism (namely, uniform default left-to-right evalaution). The improved empirical coverage concerns the interaction of pied-piping, *wh*-binding, and weak crossover.

Weak crossover is the name for the fact that a quantificational NP must usually precede any pronoun that it binds:

(48) a. Everyone$_i$ loves his$_i$ mother.
b. *His$_i$ mother loves everyone$_i$.

We claim that weak crossover follows from the assumption that by default, people evaluate expressions from left to right, combined with the fact that a quantifier must be evaluated before evaluting any pronoun that it binds.

Variable binding, of course, is a prototypical side-effect in computer programming languages. Since continuations model side-effects, binding is an ideal arena in which to test the utility of continuations for describing natural language.

Binding in general poses a problem for TLG. Standard multimodal TLG is linear, but binding by its very nature involves using a resource more than once: once for the antecedent and then again for the bound pronoun.[10]

Jäger (2001) proposes to deal with this situation by adding a new logical connective '|' and special (non-linear) inferences rules |E and |I (see page 105) for binding. Pronouns have category $np|np$; more generally, $X|Y$ means 'I function as something of category X, and I contain something of category Y that needs to be bound'. The general binding inference rule allows any expression of category Y to bind into the $X|Y$ as long as the binder precedes the bound expression.

One reason why Jäger's system is especially appropriate to consider here is that he explicitly recognizes the role of linear order in restricting binding, as we do, but unlike most LF-based accounts. Jäger's binding rule requires that the antecedent precede the pronoun it binds, and he gives empirical arguments that binding is sensitive to linear order. In particular, he correctly predicts weak crossover facts like those in (48). But he merely stipulates linear order: it follows from nothing, and the rule could just as easily have required that the binder follow its bound pronoun. Of course, Jäger is primarily interested in the role of contraction (duplication of resources), not linear order; but we would like to suggest that continuations can provide a deeper explanation for the role of linear order in crossover. On our account, the linear prohibition on weak crossover in quantificational binding follows from assuming that by default, people evaluate expressions from left to right.

In part to help compare our approach with Jäger's (2001), we also accomplish binding by means of a (single) non-linear inference rule. However, since (unlike Jäger) we are working within a multimodal TLG, rather than introducing a new logical connective |, we introduce a new modality $\circ_b$ for expressing binding relationships ('b' for 'binding'):

(49) $$\text{she} \vdash (np\backslash_b A)/\!\!/(np\backslash\!\!\backslash A).$$

[10] Dowty (1992) discusses binding and weak crossover in some early versions of TLG. In more recent but as yet unpublished work, he develops those ideas into an approach on which the duplication of resources is encapsulated entirely within the lexical meaning of the pronoun. As a result, the non-lexical part of the system, i.e., the logic proper, can remain linear. Like us, Dowty treats pronouns as quantificational, though he limits pronouns to taking scope only over types corresponding to verb phrases.

Conceptually, $np\backslash_b A$ is the category of an expression that would otherwise count as an A, but which contains a bindable pronoun. Thus the lexical entry for *she* turns any enclosing expression of category A into something of category $np\backslash_b A$. For instance, *John thought she left* would have category $np\backslash_b s$.

Our binding rule is a non-linear structural rule.[11]

(50) $$B \circledcirc K \vdash B \circ_b (B \circledcirc K) \quad \text{(Bind)}$$

It says that a subexpression B within a context $\mathcal{K}$ can bind into the larger expression $\mathcal{K}[B]$. We illustrate here by showing how to derive *Everyone$_i$ saw his$_i$ mother*, in which the subexpression *everyone* binds into *[everyone] saw his mother*.

(51)
$$
\dfrac{\text{everyone} \vdash s \mathbin{/\!\!/} (np \mathbin{\backslash\!\!\backslash} s) \quad
\dfrac{\dfrac{\dfrac{\dfrac{}{np \vdash np}\,\text{Id} \quad
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{he} \vdash (np\backslash_b s) \mathbin{/\!\!/} (np \mathbin{\backslash\!\!\backslash} s) \quad
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\\ np \circ (\text{saw} \circ (np \circ \text{’s mother})) \vdash s\end{array}}{\Diamond(np \circ (\text{saw} \circ (np \circ \text{’s mother}))) \vdash \Diamond s}\,\Diamond\text{I} \quad \dfrac{\dfrac{}{s \vdash s}\,\text{Id}}{\Diamond s \vdash s}\,\text{Unquote}}{\Diamond(np \circ (\text{saw} \circ (np \circ \text{’s mother}))) \vdash s}\,\Diamond\text{E}}{\Diamond np \circ \Diamond(\text{saw} \circ (np \circ \text{’s mother})) \vdash s}\,\text{K}'}{\Diamond np \circ (\Diamond\text{saw} \circ \Diamond(np \circ \text{’s mother})) \vdash s}\,\text{K}'}{\Diamond np \circ (\Diamond\text{saw} \circ (np \circ \text{’s mother})) \vdash s}\,\text{T}}{(\Diamond np \circ (\Diamond\text{saw} \circ (np \circ \text{’s mother}))) \circledcirc 1 \vdash s}\,\text{Push}}{(\Diamond\text{saw} \circ (np \circ \text{’s mother})) \circledcirc (1 \circ \Diamond np) \vdash s}\,\text{Right}}{(np \circ \text{’s mother}) \circledcirc ((1 \circ \Diamond np) \circ \Diamond\text{saw}) \vdash s}\,\text{Right}}{np \circledcirc (\text{’s mother} \circ ((1 \circ \Diamond np) \circ \Diamond\text{saw})) \vdash s}\,\text{Left}}{\text{’s mother} \circ ((1 \circ \Diamond np) \circ \Diamond\text{saw}) \vdash np \mathbin{\backslash\!\!\backslash} s}\,\backslash\!\!\backslash\text{I}}
{\text{he} \circledcirc (\text{’s mother} \circ ((1 \circ \Diamond np) \circ \Diamond\text{saw})) \vdash np\backslash_b s}\,/\!\!/\text{E}}
{(\text{he} \circ \text{’s mother}) \circledcirc ((1 \circ \Diamond np) \circ \Diamond\text{saw}) \vdash np\backslash_b s}\,\text{Left}}
{(\Diamond\text{saw} \circ (\text{he} \circ \text{’s mother})) \circledcirc (1 \circ \Diamond np) \vdash np\backslash_b s}\,\text{Right}}
{(\Diamond np \circ (\Diamond\text{saw} \circ (\text{he} \circ \text{’s mother}))) \circledcirc 1 \vdash np\backslash_b s}\,\text{Right}}
{(np \circ (\Diamond\text{saw} \circ (\text{he} \circ \text{’s mother}))) \circledcirc 1 \vdash np\backslash_b s}\,\text{T}}
{(np \circ (\text{saw} \circ (\text{he} \circ \text{’s mother}))) \circledcirc 1 \vdash np\backslash_b s}\,\text{T}}
{np \circledcirc ((\text{saw} \circ (\text{he} \circ \text{’s mother})) \circ 1) \vdash np\backslash_b s}\,\text{Left}}
{np \circ_b (np \circledcirc ((\text{saw} \circ (\text{he} \circ \text{’s mother})) \circ 1)) \vdash s}\,\backslash_b\text{E}}
{np \circledcirc ((\text{saw} \circ (\text{he} \circ \text{’s mother})) \circ 1) \vdash s}\,\text{Bind}}
{(\text{saw} \circ (\text{he} \circ \text{’s mother})) \circ 1 \vdash np \mathbin{\backslash\!\!\backslash} s}\,\backslash\!\!\backslash\text{I}}
{\dfrac{\dfrac{\text{everyone} \circledcirc ((\text{saw} \circ (\text{he} \circ \text{’s mother})) \circ 1) \vdash s}{(\text{everyone} \circ (\text{saw} \circ (\text{he} \circ \text{’s mother}))) \circledcirc 1 \vdash s}\,\text{Left}}{\text{everyone} \circ (\text{saw} \circ (\text{he} \circ \text{’s mother})) \vdash s}\,\text{Pop}}\,/\!\!/\text{E}
$$

A couple of points about this derivation: on this analysis, the pronoun itself is a scope-taking expression, taking scope over its binder's context. In this example, *everyone* binds *he*, and the context of *everyone* is *[ ] saw his mother*. (This context appears at the bottom of the derivation, where *everyone* enters, as $(\text{saw} \circ (\text{he} \circ \text{’s mother})) \circ 1$.) Thus *he* must take scope over this context—that is, must take as *its* context *np saw [ ]'s mother*, in which the type np plays the role of the placeholder variable bound by the quantifier *everyone*. (This context appears in the derivation, where *he* enters, as $\text{’s mother} \circ ((1 \circ \Diamond np) \circ \Diamond\text{saw})$.)

It is always easier to explain why a good derivation works than why some other sentence has no derivation, but let us offer a few words on why the crossover example **His$_i$ mother saw everyone$_i$* (48b) does not have a bound reading. The crucial difference between (48a) and (48b) is that *everyone* is now to the right of the pronoun it is trying to bind. In order for *everyone* to take scope over the context *his mother saw [ ]*, we must apply the Right postulate twice (just as in the inverse-scope derivation above in (19)). But the Right postulate requires the introduction of at least two $\Diamond$s. Once diamonds enter the picture, the only way to remove them and unwrap the content of the quoted elements is by applying the Unquote postulate. But since Unquote only applies to expressions of type s, and there is no way to arrive at the type s without first unwrapping the pronoun, there is an unresolvable standoff: once the non-unquotable $np\backslash_b s$ gets quoted, the derivation is doomed.

7.2. **Questions and superiority.** As mentioned above, our analysis provides an explanation for superiority that parallels our explanation for weak crossover in relying on left-to-right evaluation order. Superiority is the name for the fact that a *wh*-trace must usually precede any additional, in-situ *wh*-phrase:

(52) a. Who saw what?
b. Who do you think __ saw what?
c. *What did who see __?

We suggest that in (52c), the presence of the additional *wh*-phrase *who* interferes with the ability of the fronted *wh*-phrase to bind its trace. The crucial assumption is that the *wh*-trace in (52c) can only get bound if it takes outermost scope over any additional *wh*-phrase. Once again, our basic strategy will be to show that the *wh*-trace can take scope over the in-situ *wh*-phrase only if it comes first.

Of course, before we can explain superiority, we have to first provide basic lexical entries for *wh*-words, and treat their usage both in-situ and raised. Just as we introduced a new modality $\circ_b$ above to express binding relationships, we now add a new modality $\circ_?$ to express questions:

(53) $$\text{who} \vdash (np\backslash_? S) \mathbin{/\!\!/} (np \mathbin{\backslash\!\!\backslash} S),$$

(54) $$\text{what} \vdash (np\backslash_? S) \mathbin{/\!\!/} (np \mathbin{\backslash\!\!\backslash} S),$$

(55) $$\text{which} \vdash ((np\backslash_? S) \mathbin{/\!\!/} (np \mathbin{\backslash\!\!\backslash} S))/(np\backslash_n s).$$

Here the type variable S ranges over all types. Conceptually, $np\backslash_? S$ is the category of an S-question that seeks an np-answer. For example, a single-*wh* question has the type $np\backslash_? s$, and a double-*wh* question has the type $np\backslash_? np\backslash_? s$. The lexical entries above specify that *wh*-words take scope in situ, so they by themselves generate in-situ *wh*-questions. To generate raised-*wh* questions, we add two postulates.[12]

(56) $$A \circ_? (B \circ (C \circledcirc K)) \vdash A \circ_? (B \circledcirc (C \circ K)) \qquad \text{(Trace Left)}$$

(57) $$A \circ_? (C \circ (\Diamond B \circledcirc K)) \vdash A \circ_? (C \circledcirc (K \circ \Diamond B)) \qquad \text{(Trace Right)}$$

[11] If multimodal TLG supported not only multiplicative connectives but also exponentials in the linear-logic sense, we would replace the category np with the category $!_b np$ throughout our grammar (where $!_b$ is an exponential modality with respect to $\circ_b$; hence $!_b A \vdash A$ (dereliction) and $!_b A \vdash !_b A \circ_b !_b A$ (contraction)), and use the following binding rule, which is linear: $(B \circ_b C) \circledcirc K \vdash B \circ_b (C \circledcirc K)$.

[12] These two postulates are really the composition of the Left and Right postulates with a single new postulate

$$A \circ_? (B \circ (\_\_ \circledcirc K)) \vdash A \circ_? (B \circledcirc K) \quad \text{(Trace)},$$

where __ is the empty string, in other words the identity for the default mode $\circ$. But as §5 explained, the default mode should not have an identity if we want *very* to have the type $(n/n)/(n/n)$. If multimodal TLG were to allow empty antecedents everywhere and use unary modalities to enforce non-emptiness when necessary, then we would be able to replace Trace Left and Trace Right with this Trace rule—an appealing prospect.

The following derivation shows the basic usage scenario for these additions to the grammar. On one hand, the entire derivation culminates in the pied-piped raised-*wh* question *Whose mother did Alice see?*. On the other hand, the $/\!\!/$E rule near the top concludes already with the in-situ *wh*-question *Alice saw whose mother?*.

(58)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{np \vdash np}\,\text{Id} \qquad \dfrac{\text{who} \vdash (np\backslash_? s)/\!\!/(np\backslash\!\!\backslash s) \quad \begin{array}{c}\vdots\ \text{like the top of (51)}\\ \text{'s mother} \circ ((1 \circ \Diamond\text{Alice}) \circ \Diamond\text{saw}) \vdash np\backslash\!\!\backslash s\end{array}}{\text{who} \circledcirc (\text{'s mother} \circ ((1 \circ \Diamond\text{Alice}) \circ \Diamond\text{saw})) \vdash np\backslash_? s}\,/\!\!/\text{E}}{np \circ_? (\text{who} \circledcirc (\text{'s mother} \circ ((1 \circ \Diamond\text{Alice}) \circ \Diamond\text{saw}))) \vdash s}\,\backslash_?\text{E}}{np \circ_? ((\text{who} \circ \text{'s mother}) \circledcirc ((1 \circ \Diamond\text{Alice}) \circ \Diamond\text{saw})) \vdash s}\,\text{Left}}{np \circ_? ((\text{who} \circ \text{'s mother}) \circ (\Diamond\text{saw} \circledcirc (1 \circ \Diamond\text{Alice}))) \vdash s}\,\text{Trace Right}}{(\text{who} \circ \text{'s mother}) \circ (\Diamond\text{saw} \circledcirc (1 \circ \Diamond\text{Alice})) \vdash np\backslash_? s}\,\backslash_?\text{I}}{(\text{who} \circ \text{'s mother}) \circ ((\Diamond\text{Alice} \circ \Diamond\text{saw}) \circledcirc 1) \vdash np\backslash_? s}\,\text{Right}}{(\text{who} \circ \text{'s mother}) \circ (\Diamond\text{Alice} \circ \Diamond\text{saw}) \vdash np\backslash_? s}\,\text{Pop}}{(\text{who} \circ \text{'s mother}) \circ (\text{Alice} \circ \Diamond\text{saw}) \vdash np\backslash_? s}\,\text{T}}{(\text{who} \circ \text{'s mother}) \circ (\text{Alice} \circ \text{saw}) \vdash np\backslash_? s}\,\text{T}$$

Of course, the result should be *Whose mother did Alice see?*, not *Whose mother Alice saw?*. We ignore subject-auxiliary inversion and verb forms purely for the sake of keeping the derivations simpler.

Let us now examine the derivation of a double-*wh* question that abides by superiority, so that we can see what goes wrong in an attempt to violate superiority. Below we derive the question *Who saw what?*, in which *who* is raised and *what* remains in situ. (We say that *who* is raised, even though it does not affect the string, because the derivation uses Trace Left near the end. We use *Who saw what?* as our example for simplicity.)

(59)

$$\dfrac{\dfrac{\dfrac{\dfrac{}{np \vdash np}\,\text{Id} \qquad \dfrac{\text{who} \vdash (np\backslash_?(np\backslash_? s))/\!\!/(np\backslash\!\!\backslash(np\backslash_? s)) \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{what} \vdash (np\backslash_? s)/\!\!/(np\backslash\!\!\backslash s) \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{np \circ (\text{saw} \circ np) \vdash s}\,(14)}{\Diamond(np \circ (\text{saw} \circ np)) \vdash \Diamond s}\,\Diamond\text{I} \quad \dfrac{\dfrac{}{s \vdash s}\,\text{Id}}{\Diamond s \vdash s}\,\text{Unquote}}{\Diamond(np \circ (\text{saw} \circ np)) \vdash s}\,\Diamond\text{E}}{\Diamond np \circ \Diamond(\text{saw} \circ np) \vdash s}\,\text{K}'}{\Diamond np \circ (\Diamond\text{saw} \circ \Diamond np) \vdash s}\,\text{K}'}{\Diamond np \circ (\Diamond\text{saw} \circ np) \vdash s}\,\text{T}}{(\Diamond np \circ (\Diamond\text{saw} \circ np)) \circledcirc 1 \vdash s}\,\text{Push}}{(\Diamond\text{saw} \circ np) \circledcirc (1 \circ \Diamond np) \vdash s}\,\text{Right}}{np \circledcirc ((1 \circ \Diamond np) \circ \Diamond\text{saw}) \vdash s}\,\text{Right}}{(1 \circ \Diamond np) \circ \Diamond\text{saw} \vdash np\backslash\!\!\backslash s}\,\backslash\!\!\backslash\text{I}}{\text{what} \circledcirc ((1 \circ \Diamond np) \circ \Diamond\text{saw}) \vdash np\backslash_? s}\,/\!\!/\text{E}}{(\Diamond\text{saw} \circ \text{what}) \circledcirc (1 \circ \Diamond np) \vdash np\backslash_? s}\,\text{Right}}{(\Diamond np \circ (\Diamond\text{saw} \circ \text{what})) \circledcirc 1 \vdash np\backslash_? s}\,\text{Right}}{\Diamond np \circledcirc ((\Diamond\text{saw} \circ \text{what}) \circ 1) \vdash np\backslash_? s}\,\text{Left}}{np \circledcirc ((\Diamond\text{saw} \circ \text{what}) \circ 1) \vdash np\backslash_? s}\,\text{T}}{np \circledcirc ((\text{saw} \circ \text{what}) \circ 1) \vdash np\backslash_? s}\,\text{T}}{(\text{saw} \circ \text{what}) \circ 1 \vdash np\backslash\!\!\backslash(np\backslash_? s)}\,\backslash\!\!\backslash\text{I}}{\text{who} \circledcirc ((\text{saw} \circ \text{what}) \circ 1) \vdash np\backslash_?(np\backslash_? s)}\,/\!\!/\text{E}}{np \circ_? (\text{who} \circledcirc ((\text{saw} \circ \text{what}) \circ 1)) \vdash np\backslash_? s}\,\backslash_?\text{E}}{np \circ_? (\text{who} \circ ((\text{saw} \circ \text{what}) \circledcirc 1)) \vdash np\backslash_? s}\,\text{Trace Left}}{\dfrac{\text{who} \circ ((\text{saw} \circ \text{what}) \circledcirc 1) \vdash np\backslash_?(np\backslash_? s)}{\text{who} \circ (\text{saw} \circ \text{what}) \vdash np\backslash_?(np\backslash_? s)}\,\text{Pop}}\,\backslash_?\text{I}$$

In the derivation above, Trace Left applies to the raised *wh*-phrase *who* in the context *[ ] saw what*. In other words, the raised *wh*-phrase takes scope over the rest of the sentence. Similarly, to derive the superiority violation **What did who see __?* (or **what who saw*, to ignore subject-auxiliary inversion), *what* must take scope over the context *who saw [ ]*. Just as in the previous section, taking scope over such a context requires using Right postulate twice, which introduces two ◇s and necessitates Unquote. But questions cannot be unquoted: their types are of the form $A\backslash_? S$, not $\Diamond_u S$. Superiority is thus enforced.

For Jäger, *wh*-phrases have category $q/(np \uparrow s)$, where q is the category of questions and "$np \uparrow s$" is an s with an np-gap. Because connecting gaps with their fillers uses a mechanism separate from binding, whatever might explain superiority in Jäger's system will be independent of the explanation for weak crossover. On our account, connecting a *wh*-trace with its filler uses the same continuation-based mechanism as binding, and the explanation for why an intervening in-situ *wh*-word prevents a fronted *wh*-phrase from binding its trace follows from the same evaluation order mechanism as weak crossover.

Whether or not weak crossover and superiority ought to have a unified explanation is debatable, and further investigation may reveal empirical arguments that these two phenomena are in fact distinct. We will not attempt to settle that question here; our main point is that we cannot even have the debate in TLG unless continuations are made available.

As for empirical coverage, there is an interesting interaction between *wh*-binding and weak crossover that we call *pied binding*, which seems to favor our unified treatment.

(60) a. Who$_i$ __ saw his$_i$ mother?
b. *Who$_i$ did his$_i$ mother see __?

To a first approximation, a *wh*-phrase can bind a pronoun only if the trace of the *wh*-phrase precedes the pronoun. One explanation that has been popular at least since Reinhart (1983) is that the trace itself does the binding. On Jäger's system, the trace can indeed accomplish the binding in examples such as (60a) (see especially pages 124–125). The unacceptability of (60b) is also explained, since when the trace follows the pronoun, the prohibition against cataphora predicts the relative unacceptability of (60b).

But there are more complex examples which cast doubt on the assumption that it is the trace that is doing the binding:

(61) a. Whose$_i$ father did John say __ saw his$_i$ mother?
b. *Whose$_i$ father did John say his$_i$ mother saw __?

In each case, the trace is bound by the entire pied-piped *wh*-phrase *whose father*. That is, in each case, it is a father who is seeing or being seen. Yet the pronoun can be bound by the *wh*-word alone, as indicated by the indexation in (61a), suggesting that the *wh*-word must be able to bind the pronoun directly after all.

Interestingly, (61b) shows that binding still requires linear precedence of a sort: the pronoun can be bound by the *wh*-word *whose* only if the *wh*-trace precedes the pronoun. In our system, *whose* can bind a pronoun just as a quantificational NP does. But just as superiority is violated when an in-situ *wh*-phrase interferes between a raised *wh*-phrase and its trace, weak crossover is perpetrated when a bindable pronoun interferes between a raised *wh*-phrase and its trace. Put differently, (61b) constitutes a weak crossover violation because the corresponding in-situ *wh*-question *John said his$_i$ mother saw whose$_i$ father?* does. This complex pattern of interactions between pied-piping, binding of *wh*-traces, and binding of pronouns falls out from our unified account without additional stipulation.

This brief discussion hardly counts as a thorough comparison of Shan and Barker 2003 with Jäger 2001; that would take far too much space here. Our only intention is to suggest that a continuation-based analysis of a complex phenomenon can hold its own against a robust previous TLG account, and to point out that our account provides a different pattern of explanation. In particular, we connect the linear order asymmetry of weak crossover with the computational concept of evaluation order; and we also provide an analysis on which weak crossover and superiority have distinct but closely parallel explanations. Thus continuations constitute a new and potentially valuable explanatory tool for TLG analyses.

7.3. **Polarity licensing.** Polarity sensitivity (Ladusaw 1979) has been a popular linguistic phenomenon to analyze in the type-logical (Bernardi 2002; Bernardi and Moot 2001), categorial (Dowty 1994), and lexical-functional (Fry 1997, 1999) approaches to grammar. The multitude of these analyses is in part due to the more explicit emphasis that these traditions place on the syntax-semantics interface—be it in the form of the Curry-Howard isomorphism, Montague-style semantic rules, or linear logic as glue—and the fact that polarity sensitivity is a phenomenon that spans syntax and semantics.

On one hand, which polarity items are licensed or prohibited in a given linguistic environment depends, by and large, on semantic properties of that environment (Krifka 1995; Ladusaw 1979, inter alia). For example, to a first approximation, negative polarity items can occur only in *downward-entailing* contexts, such as under the scope of a *monotonically decreasing* quantifier. A quantifier q, of type $(np \to s) \to s$, is monotonically decreasing

just in case

(62)
$$\forall s_1.\, \forall s_2.\, (\forall x.\, s_2(x) \Rightarrow s_1(x)) \Rightarrow q(s_1) \Rightarrow q(s_2).$$

Thus (63a) is acceptable because the scope of *no one* is downward-entailing, whereas (63b) and (63c) are unacceptable.

(63) a. No one saw anyone.
b. *Everyone saw anyone.
c. *Alice saw anyone.

To account for these contrasts, Fry (1997, 1999), Bernardi (2002), and Bernardi and Moot (2001) all split the clause type s into several types, related by subtyping. Fry (1997, 1999) distinguishes between s and $\ell \multimap (s \otimes \ell)$ in linear logic, where ℓ stands for polarity licensing as a grammatical resource. Analogously, Bernardi (2002) and Bernardi and Moot (2001) distinguish between different unary modalities applied to s.

A simplistic version of Bernardi's analysis is to distinguish between the clause types s ('neutral clause') and $\Box_p^\downarrow\Diamond_p s$ ('negative clause'). Here $\Diamond_p$ is a new unary mode (the letter p stands for "polarity"), and we abbreviate $\Box_p^\downarrow\Diamond_p s$ to s^-. We choose the type $\Box_p^\downarrow\Diamond_p s$ so that $s \vdash s^-$ is a theorem in multimodal TLG.

(64)
$$\dfrac{\dfrac{}{\Diamond_p s \vdash \Diamond_p s}\,\text{Id}}{s \vdash \Box_p^\downarrow\Diamond_p s}\,\Box_p^\downarrow\text{I}$$

Splitting the clause type like this helps us account for polarity sensitivity, because we can now assign different types to different quantifiers in the lexicon:

(65)
$$\text{everyone} \vdash s /\!\!/ (np \backslash\!\!\backslash s), \qquad \text{no one} \vdash s /\!\!/ (np \backslash\!\!\backslash s^-), \qquad \text{anyone} \vdash s^- /\!\!/ (np \backslash\!\!\backslash s^-).$$

The type of *everyone* is unchanged: it takes scope over a neutral clause to form a neutral clause. The types of *no one* and *anyone* involve the newly introduced s^-: they both take scope over a negative clause, but *no one* forms a neutral clause whereas *anyone* forms a negative clause. As (64) shows, a neutral clause can be converted to a negative clause. Thus, for example, *anyone* can take scope in $np\circ(\text{saw}\circ\text{anyone})$ to form a negative clause s^-, even though the verb *saw* produces a neutral clause s initially.

(66)
$$\dfrac{\dfrac{\dfrac{\text{anyone} \vdash s^- /\!\!/ (np \backslash\!\!\backslash s^-) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{np \circ (\text{saw} \circ np) \vdash s}\,(14)}{\Diamond_p(np \circ (\text{saw} \circ np)) \vdash \Diamond_p s}\,\Diamond_p\text{I}}{np \circ (\text{saw} \circ np) \vdash s^-}\,\Box_p^\downarrow\text{I}}{(np \circ (\text{saw} \circ np)) \circledcirc 1 \vdash s^-}\,\text{Push}}{(\text{saw} \circ np) \circledcirc (1 \circ np) \vdash s^-}\,\text{Right}}{np \circledcirc ((1 \circ np) \circ \text{saw}) \vdash s^-}\,\text{Right}}{(1 \circ np) \circ \text{saw} \vdash np \backslash\!\!\backslash s^-}\,\backslash\!\!\backslash\text{I}}{\text{anyone} \circledcirc ((1 \circ np) \circ \text{saw}) \vdash s^-}\,/\!\!/\text{E}}{(\text{saw} \circ \text{anyone}) \circledcirc (1 \circ np) \vdash s^-}\,\text{Right}}{(np \circ (\text{saw} \circ \text{anyone})) \circledcirc 1 \vdash s^-}\,\text{Right}}{np \circ (\text{saw} \circ \text{anyone}) \vdash s^-}\,\text{Pop}$$

As before, we consider a derivation complete if it culminates in the type s, not s^-. The fact that $np \circ (\text{saw} \circ \text{anyone})$ only has the type s^- (as derived above) and not s predicts that a clause like **Alice saw anyone* (63c) is not a grammatical sentence by itself. Moreover, because there is no way to convert a negative clause to a neutral clause, *everyone* cannot take scope over a negative clause, so **Everyone saw anyone* (63b) is ruled out as well. However, *no one* (unlike *Alice* or *everyone*) can take scope over *anyone* to form a complete (neutral) clause. Thus even our simplistic rendering of Bernardi's account predicts correctly that *No one saw anyone* (63a) has a linear-scope reading (but no inverse-scope reading). We also predict correctly that the sentence

(67) No one introduced everyone to anyone.

has no linear-scope reading, even though it is acceptable when *no one* scopes over *anyone*, then *everyone*. (Szabolcsi (2004) discusses such "intervention" cases.)

On the other hand, a restriction on surface syntactic form, such as that imposed by polarity sensitivity, is by definition a matter of syntax. Besides, there are syntactic restrictions on the configuration relating the licensor to the licensee. For example, (63a) above is acceptable—*no one* manages to license *anyone*—but (68) below is not. As the contrasts in (69) and (70) further illustrate, the licensor usually needs to precede the licensee.

(68) *Anyone saw no one.

(69) a. Nobody's mother saw anybody's father.
b. *Anybody's mother saw nobody's father.

(70) a. Alice didn't visit anyone.
b. *Anyone didn't visit Alice.

These and other examples show that the syntactic relations allowed between licensor and licensee for polarity purposes are similar to those allowed between antecedent and pronoun for binding purposes. Because Fry, Bernardi, and Bernardi and Moot focus on quantification and scope, they easily characterize how *no* must scope over *any* in order to license it, but they leave it a mystery why *no* must precede *any* in order to license it. In particular, they wrongly accept all of (68–70). This mystery has been noted by Ladusaw (1979, §9.2), and Fry (1999, §8.2) likewise identifies it as a defect in current accounts of polarity sensitivity.

Continuations provide precisely the missing link between linear order and quantifier scope. Recall from §6.3 that inverse scope can be generated, even under a regime of left-to-right evaluation, using the Unquote rule in (46). The crucial step, as illustrated in (47), is to unquote the clause produced by the narrower-scope quantifier. In that example, *everyone* can take inverse scope over *someone*, because *someone* produces a neutral clause s. A neutral clause can be unquoted because its type s is shorthand for $\Diamond_u s'$, which is enclosed in $\Diamond_u$. By contrast, a negative clause cannot be unquoted because its type s^- is shorthand for $\Box_p^{\downarrow}\Diamond_p\Diamond_u s'$, which is not enclosed in $\Diamond_u$. Given that *anyone* produces s^-, then, inverse scope over *anyone* is impossible. This prediction correctly rules out the unacceptable examples in (68–70) while leaving (67) available.

This explanation for linear-order effects in polarity licensing is further developed elsewhere (Shan 2004). We are not aware of any other account of polarity sensitivity that unifies its syntactic properties with weak crossover as we do using left-to-right evaluation.

7.4. **Reversing evaluation order.** One way to see that our implementation of evaluation order unifies linear order effects in weak crossover, superiority, and polarity is to impose right-to-left evaluation and see what happens. Because our Left and Right postulates embody left-to-right evaluation order, we can reverse default evaluation order while leaving all other aspects of the system unchanged.

More specifically, suppose that we replace the Left and Right postulates with

(71) $B \circledcirc (\Diamond C \circ K) \dashv\vdash (B \circ \Diamond C) \circledcirc K$ (Left), $(B \circ C) \circledcirc K \dashv\vdash C \circledcirc (K \circ B)$ (Right).

We can still derive linear and inverse scope for quantifiers, but the predictions concerning the weak crossover judgments in (48), the superiority judgments in (52), and the polarity judgments in (68–70) reverse: binders must follow any pronouns they bind, *wh*-traces must follow any additional *wh*-phrases, and polarity triggers must follow the polarity items they license.

8. Conclusions

Let's take stock. Our core proposal implements the full power of the quantificational type constructor q using only a single new mode ($\circledcirc$) along with two structural postulates, Left and Right. (If empty antecedents are forbidden, we also need to simulate Push and Pop.) We provided a geometric interpretation of types to explain how this parsimonious reduction of q constitutes an implementation of continuations: under the geometric interpretation, the continuation mode decomposes types into a subexpression in the foreground and a remainder in the background, where the remainder is interpreted as a context.

We then (§6.3) imposed left-to-right evaluation order by stipulating that a quantificational element could only take scope over an expression to its left if that expression had no side effects, i.e., was "pure", as indicated by the presence of the unary modality $\Diamond$. In order to restore inverse scope, we added an Unquote rule to remove the $\Diamond$, though only for expressions in category s. Under the analogy with computation, the evaluation of an expression is delayed for as long as it is under the quotation modality $\Diamond$. By quoting a leftmost quantifier and then unquoting it later, we can delay the evaluation of the quantifier and let it take narrow (hence inverse) scope with respect to some following quantifier.

We proposed analyses on which pronominal binding (§7.1), the binding of a *wh*-trace (§7.2), and licensing of a negative polarity item (§7.3) were all accomplished via side-effects. Because bindable pronouns have their own side-effects, their presence disrupted the unquotation strategy, so that the analysis automatically predicts that a quantifier can take scope over a pronoun to its left but cannot bind it. That is, the analysis correctly allows inverse scope but not weak crossover violations. Similarly, the analysis predicts that a *wh*-phrase can bind its trace only when no other *wh*-phrase intervenes. That is, the analysis correctly allows multiple *wh*-questions but not superiority violations. Finally, the analysis automatically correctly requires a negative polarity licensor to precede its licensee.

We believe that the theoretical parsimony and empirical robustness of these analyses show how continuations offer new and valuable tools for multimodal type-logical grammar writers. In particular, continuation-based analyses offer new insights into quantification, binding, multiple *wh*-questions, and negative polarity.

References

Aoun, Joseph, and Yen-hui Audrey Li. 1993. *Syntax of scope*. Cambridge: MIT Press.

Barendregt, Henk P. 1981. *The lambda calculus: Its syntax and semantics*. Amsterdam: Elsevier Science.

Barker, Chris. 2002a. Continuations and the nature of quantification. *Natural Language Semantics* 10(3):211–242.

———. 2002b. Remark on Jacobson 1999: Crossover as a local constraint. *Linguistics and Philosophy*. To appear.

———. 2004. Continuations in natural language (extended abstract). In Thielecke (2004), 1–11.

Bernardi, Raffaella. 2002. Reasoning with polarity in categorial type logic. Ph.D. thesis, Utrecht Institute of Linguistics (OTS), Utrecht University.

———. 2003. CTL: In situ binding. `http://www.let.uu.nl/~Raffaella.Bernardi/personal/q_solutions.pdf`.

Bernardi, Raffaella, and Richard Moot. 2001. Generalized quantifiers in declarative and interrogative sentences. *Journal of Language and Computation* 1(3):1–19.

Dalrymple, Mary, ed. 1999. *Semantics and syntax in Lexical Functional Grammar: The resource logic approach*. Cambridge: MIT Press.

Dalrymple, Mary, John Lamping, Fernando Pereira, and Vijay A. Saraswat. 1999. Quantification, anaphora, and intensionality. In Dalrymple (1999), chap. 2, 39–89.

Danvy, Olivier, and Andrzej Filinski. 1989. A functional abstraction of typed contexts. Tech. Rep. 89/12, DIKU, University of Copenhagen, Denmark. `http://www.daimi.au.dk/~danvy/Papers/fatc.ps.gz`.

Dowty, David. 1992. 'Variable-free' syntax, variable-binding syntax, the natural deduction Lambek calculus, and the crossover constraint. In *Proceedings of the 11th West Coast Conference on Formal Linguistics*, ed. Jonathan Mead. Stanford, CA: Center for the Study of Language and Information.

Dowty, David R. 1994. The role of negative polarity and concord marking in natural language reasoning. In *SALT IV: Semantics and linguistic theory*, ed. Mandy Harvey and Lynn Santelmann. Ithaca: Cornell University Press.

Felleisen, Matthias. 1988. The theory and practice of first-class prompts. In *POPL '88: Conference record of the annual ACM symposium on principles of programming languages*, 180–190. New York: ACM Press.

Fry, John. 1997. Negative polarity licensing at the syntax-semantics interface. In *Proceedings of the 35th annual meeting of the Association for Computational Linguistics and 8th conference of the European chapter of the Association for Computational Linguistics*, ed. Philip R. Cohen and Wolfgang Wahlster, 144–150. San Francisco: Morgan Kaufmann.

———. 1999. Proof nets and negative polarity licensing. In Dalrymple (1999), chap. 3, 91–116.

de Groote, Philippe. 2001. Type raising, continuations, and classical logic. In *Proceedings of the 13th Amsterdam Colloquium*, ed. Robert van Rooy and Martin Stokhof, 97–101. Institute for Logic, Language and Computation, Universiteit van Amsterdam.

Huang, Cheng-Teh James. 1982. Logical relations in Chinese and the theory of grammar. Ph.D. thesis, Department of Linguistics and Philosophy, Massachusetts Institute of Technology.

Jacobson, Pauline. 1999. Towards a variable-free semantics. *Linguistics and Philosophy* 22(2):117–184.

Jäger, Gerhard. 2001. Anaphora and type logical grammar. Habilitationsschrift, Humboldt University Berlin. UIL-OTS Working Papers 01004-CL/TL, Utrecht Institute of Linguistics (OTS), Utrecht University.

Krifka, Manfred. 1995. The semantics and pragmatics of polarity items. *Linguistic Analysis* 25:209–257.

Ladusaw, William A. 1979. Polarity sensitivity as inherent scope relations. Ph.D. thesis, Department of Linguistics, University of Massachusetts. Reprinted by New York: Garland, 1980.

Lambek, Joachim. 1958. The mathematics of sentence structure. *American Mathematical Monthly* 65(3):154–170.

Meyer, Albert R., and Mitchell Wand. 1985. Continuation semantics in typed lambda-calculi (summary). In *Logics of programs*, ed. Rohit Parikh, 219–224. Lecture Notes in Computer Science 193, Berlin: Springer-Verlag.

Moortgat, Michael. 1988. *Categorial investigations: Logical and linguistic aspects of the Lambek calculus*. Dordrecht: Foris.

———. 1995. In situ binding: A modal analysis. In *Proceedings of the 10th Amsterdam Colloquium*, ed. Paul Dekker and Martin Stokhof, 539–549. Institute for Logic, Language and Computation, Universiteit van Amsterdam.

———. 1996a. Categorial type logics. In *Handbook of logic and language*, ed. Johan van Benthem and Alice ter Meulen, chap. 2. Amsterdam: Elsevier Science.

———. 1996b. Generalized quantification and discontinuous type constructors. In *Discontinuous constituency*, ed. Harry C. Bunt and Arthur van Horck, 181–207. Berlin: Mouton de Gruyter.

———. 2000. Computational semantics: Lab sessions. `http://www.let.uu.nl/~Michael.Moortgat/personal/Courses/cslab2000s.pdf`.

Moot, Richard. 2002. Proof nets for linguistic analysis. Ph.D. thesis, Utrecht Institute of Linguistics (OTS), Utrecht University.

Morrill, Glyn V. 1994. *Type logical grammar: Categorial logic of signs*. Dordrecht: Kluwer.

Papaspyrou, Nikolaos S. 1998. Denotational semantics of evaluation order in expressions with side effects. In *Recent advances in information science and technology: 2nd part of the proceedings of the 2nd IMACS international conference on circuits, systems and computers*, ed. Nikos E. Mastorakis, 87–94. Singapore: World Scientific.

Parigot, Michel. 1992. $\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction. In *Proceedings of LPAR '92: International conference on logic programming and automated reasoning*, ed. Andrei Voronkov, 190–201. Lecture Notes in Artificial Intelligence 624, Berlin: Springer-Verlag.

Reinhart, Tanya. 1983. *Anaphora and semantic interpretation*. London: Croom Helm.

Restall, Greg. 2000. *An introduction to substructural logics*. London: Routledge.

Shan, Chung-chieh. 2002. A continuation semantics of interrogatives that accounts for Baker's ambiguity. In *SALT XII: Semantics and linguistic theory*, ed. Brendan Jackson, 246–265. Ithaca: Cornell University Press.

———. 2004. Polarity sensitivity and evaluation order in type-logical grammar. In *Proceedings of the 2004 human language technology conference of the North American chapter of the Association for Computational Linguistics*, vol. 2, 129–132. Somerset, NJ: Association for Computational Linguistics.

Shan, Chung-chieh, and Chris Barker. 2003. Explaining crossover and superiority as left-to-right evaluation. Draft manuscript, Harvard University and University of California, San Diego; `http://semanticsarchive.net/Archive/TBjZDQ3Z/`.

Sitaram, Dorai, and Matthias Felleisen. 1990. Control delimiters and their hierarchies. *Lisp and Symbolic Computation* 3(1):67–99.

Steedman, Mark J. 2000. *The syntactic process*. Cambridge: MIT Press.

Szabolcsi, Anna. 1997. Strategies for scope taking. In *Ways of scope taking*, ed. Anna Szabolcsi, chap. 4, 109–154. Dordrecht: Kluwer.

———. 2004. Positive polarity—negative polarity. *Natural Language and Linguistic Theory* 22(2):409–452.

Taha, Walid, and Michael Florentin Nielsen. 2003. Environment classifiers. In *POPL '03: Conference record of the annual ACM symposium on principles of programming languages*, 26–37. New York: ACM Press.

Thielecke, Hayo, ed. 2004. *CW'04: Proceedings of the 4th ACM SIGPLAN workshop on continuations*. Tech. Rep. CSR-04-1, School of Computer Science, University of Birmingham.

Department of Linguistics, University of California, San Diego
E-mail address: barker@ucsd.edu
URL: http://www.ling.ucsd.edu/~barker/

Division of Engineering and Applied Sciences, Harvard University
E-mail address: ccshan@post.harvard.edu
URL: http://www.digitas.harvard.edu/~ken/

Polarity sensitivity and evaluation order in type-logical grammar

Chung-chieh Shan
Division of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138
ccshan@post.harvard.edu

Abstract

We present a novel, type-logical analysis of *polarity sensitivity*: how negative polarity items (like *any* and *ever*) or positive ones (like *some*) are licensed or prohibited. It takes not just scopal relations but also linear order into account, using the programming-language notions of delimited continuations and evaluation order, respectively. It thus achieves greater empirical coverage than previous proposals.

1 Introduction

Polarity sensitivity (Ladusaw, 1979) has been a popular linguistic phenomenon to analyze in the categorial (Dowty, 1994), lexical-functional (Fry, 1997, 1999), and type-logical (Bernardi, 2002; Bernardi and Moot, 2001) approaches to grammar. The multitude of these analyses is in part due to the more explicit emphasis that these traditions place on the syntax-semantics interface—be it in the form of Montague-style semantic rules, the Curry-Howard isomorphism, or linear logic as glue—and the fact that polarity sensitivity is a phenomenon that spans syntax and semantics.

On one hand, which polarity items are licensed or prohibited in a given linguistic environment depends, by and large, on semantic properties of that environment (Ladusaw, 1979; Krifka, 1995, inter alia). For example, to a first approximation, negative polarity items can occur only in *downward-entailing* contexts, such as under the scope of a *monotonically decreasing* quantifier. A quantifier q, of type $(e \to t) \to t$ where e is the type of individuals and t is the type of truth values, is monotonically decreasing just in case

(1) $\forall s_1. \forall s_2. (\forall x. s_2(x) \Rightarrow s_1(x)) \Rightarrow q(s_1) \Rightarrow q(s_2).$

Thus (2a) is acceptable because the scope of *nobody* is downward-entailing, whereas (2b–c) are unacceptable.

(2) a. Nobody saw anybody.
b. *Everybody saw anybody.
c. *Alice saw anybody.

On the other hand, a restriction on surface syntactic form, such as that imposed by polarity sensitivity, is by definition a matter of syntax. Besides, there are syntactic restrictions on the configuration relating the licensor to the licensee. For example, (2a) above is acceptable—*nobody* manages to license *anybody*—but (3) below is not. As the contrast in (4) further illustrates, the licensor usually needs to precede the licensee.

(3) *Anybody saw nobody.

(4) a. Nobody's mother saw anybody's father.
b. *Anybody's mother saw nobody's father.

The syntactic relations allowed between licensor and licensee for polarity sensitivity purposes are similar to those allowed between antecedent and pronoun for variable binding purposes. To take one example, just as an antecedent's ability to bind a (c-commanded) pronoun percolates up to a containing phrase (such as in (5), what Büring (2001) calls "binding out of DP"), a licensor's ability to license a (c-commanded) polarity item percolates up to a containing phrase (such as in (4a)).

(5) [Every boy$_i$'s mother] saw his$_i$ father.

Moreover, just as a bindee can precede a binder in a sentence when the bindee sits in a clause that excludes the binder (as in (6); see Williams, 1997, §2.1), a licensee can precede a licensor in a sentence when the licensee sits in a clause that excludes the licensor (as in (7); see Ladusaw, 1979, page 112).

(6) That he$_i$ would be arrested for speeding came as a surprise to every$_i$ motorist.

(7) That anybody would be arrested for speeding came as a surprise to the district attorney.

This paper presents a new, type-logical account of polarity sensitivity that encompasses the semantic properties exemplified in (2) and the syntactic properties exemplified in (3–4). Taking advantage of the Curry-Howard isomorphism, it is the first account of polarity sensitivity in the grammatical frameworks mentioned above to correctly fault (3) for the failure of *nobody* to appear before

$$\frac{}{A \vdash A}\ \text{Axiom}$$

For each unary mode α (blank, u, or p in this paper):

$$\frac{\Diamond_\alpha \Gamma \vdash A}{\Gamma \vdash \Box_\alpha^\downarrow A}\ \Box_\alpha^\downarrow\mathrm{I} \qquad \frac{\Gamma \vdash \Box_\alpha^\downarrow A}{\Diamond_\alpha \Gamma \vdash A}\ \Box_\alpha^\downarrow\mathrm{E}$$

$$\frac{\Gamma \vdash A}{\Diamond_\alpha \Gamma \vdash \Diamond_\alpha A}\ \Diamond_\alpha\mathrm{I} \qquad \frac{\Delta \vdash \Diamond_\alpha A \qquad \Gamma[\Diamond_\alpha A] \vdash B}{\Gamma[\Delta] \vdash B}\ \Diamond_\alpha\mathrm{E}$$

For each binary mode β (blank or c in this paper):

$$\frac{\Gamma \vdash B \quad \Delta \vdash C}{\Gamma \circ_\beta \Delta \vdash B \circ_\beta C}\ \circ_\beta\mathrm{I} \qquad \frac{\Delta \vdash B \circ C \quad \Gamma[B \circ_\beta C] \vdash A}{\Gamma[\Delta] \vdash A}\ \circ_\beta\mathrm{E}$$

$$\frac{\Delta \circ_\beta B \vdash C}{\Delta \vdash C/_\beta B}\ /_\beta\mathrm{I} \qquad \frac{\Delta \vdash B/_\beta A \qquad \Gamma \vdash A}{\Delta \circ_\beta \Gamma \vdash B}\ /_\beta\mathrm{E}$$

$$\frac{B \circ_\beta \Delta \vdash C}{\Delta \vdash B\backslash_\beta C}\ \backslash_\beta\mathrm{I} \qquad \frac{\Gamma \vdash A \qquad \Delta \vdash A\backslash_\beta B}{\Gamma \circ_\beta \Delta \vdash B}\ \backslash_\beta\mathrm{E}$$

Figure 1: Natural deduction rules for multimodal categorial grammar (Bernardi, 2002, pages 9 and 50). To reduce notation, we do not distinguish structural punctuation from logical connectives.

(Root)	$A \dashv\vdash A \circ_c 1$
(Left)	$(B \circ C) \circ_c K \dashv\vdash B \circ_c (C \circ K)$
(Right)	$(\Diamond B \circ C) \circ_c K \dashv\vdash C \circ_c (K \circ \Diamond B)$
(T)	$A \ \vdash \Diamond A$
(K′)	$\Diamond A \circ \Diamond B \ \vdash \Diamond(A \circ B)$
(Unquote)	$\Diamond\Diamond_u A \ \vdash \Diamond_u A$

Figure 2: Structural postulates

anybody. The analysis makes further correct predictions, as we will see at the end of §3.

The analysis here borrows the concepts of *delimited continuations* (Felleisen, 1988; Danvy and Filinski, 1990) and *evaluation order* from the study of programming languages. Thus this paper is about computational linguistics, in the sense of applying insights from computer science to linguistics. The basic idea transfers to other formalisms, but type-logical grammar—more precisely, multimodal categorial grammar—offers a fragment $\mathrm{NL}\Diamond_{\mathcal{R}^-}$ whose parsing problem is decidable using proof-net technology (Moot, 2002, §9.2), which is of great help while developing and testing the theory.

2 Delimited continuations

Figure 1 shows natural deduction rules for multimodal categorial grammar, a member of the type-logical family of grammar formalisms (Moortgat, 1996a; Bernardi, 2002). Figure 2 lists our structural postulates. These two figures form the logic underlying our account.

We use two binary modes: the default mode (blank) for surface syntactic composition, and the continuation mode c. As usual, a formula of the form $A \circ B$ can be read as "A followed by B". By contrast, a formula of the form $A \circ_c B$ can be read as "A in the context B". In programming-language terms, the formula $A \circ_c B$ plugs a subexpression A into a delimited continuation B. The Root rule creates a trivial continuation: it says that 1 is a right identity for the c mode, where 1 can be thought of as a nullary connective, effectively enabling empty antecedents for the c mode. The binary modes, along with the first three postulates in Figure 2, provide a new way to encode Moortgat's ternary connective q (1996b) for in-situ quantification. For intuition, it may help to draw logical formulas as binary trees, distinguishing graphically between the two modes.

To further capture the interaction between scope inversion and polarity sensitivity exemplified in (3–4), we use three unary modes: the value mode (blank), the unquotation mode u, and the polarity mode p. The value mode marks when an expression is devoid of in-situ quantification, or, in programming-language terms, when it is a pure value rather than a computation with control effects. As a special case, any formula can be turned pure by embedding it under a diamond using the T postulate, analogous to *quotation* or *staging* in programming languages. Quotations can be concatenated using the K′ postulate. The unquotation mode u marks when a diamond can be canceled using the Unquote postulate. Unquotation is also known as *eval* or *run* in programming languages. The polarity mode p, and the empirical utility of these unary modes, are explained in §3.

A derivation is considered complete if it culminates in a sequent whose antecedent is built using the default binary mode $\circ$ only, and whose conclusion is a type of the form $\Diamond_u A$. Below is a derivation of *Alice saw Bob*.

(8)
$$\dfrac{\text{Alice} \vdash np \qquad \dfrac{\text{saw} \vdash (np\backslash\Diamond_u s)/np \quad \text{Bob} \vdash np}{\text{saw} \circ \text{Bob} \vdash np\backslash\Diamond_u s}\ /\mathrm{E}}{\text{Alice} \circ (\text{saw} \circ \text{Bob}) \vdash \Diamond_u s}\ \backslash\mathrm{E}$$

Note that clauses take the type $\Diamond_u s$ rather than the usual s, so the Unquote rule can operate on clauses. We abbreviate $\Diamond_u s$ to s° below.

To illustrate in-situ quantification, Figure 3 on the following page shows a derivation of *Alice saw a man's mother*. For brevity, we treat *a man* as a single lexical item. It is a quantificational noun phrase whose polarity is *neutral* in a sense that contrasts with other quantifiers considered below. The crucial part of this derivation is the use of the structural postulates Root, Left, and Right to divide the sentence into two parts: the subexpression *a man* and its context *Alice saw _'s mother*. The type of *a man*, $s^\circ/_c(np\backslash_c s^\circ)$, can be read as "a subexpression that produces a clause when placed in a context that can enclose an np to make a clause".

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{a man} \vdash s^\circ /_c (np\backslash_c s^\circ) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{Alice} \vdash np \qquad \dfrac{\text{saw} \vdash (np\backslash s^\circ)/np \qquad \dfrac{\dfrac{}{np \vdash np}\,\text{Axiom} \qquad \text{'s mother} \vdash np\backslash np}{np \circ \text{'s mother} \vdash np}\,\backslash E}{\text{saw} \circ (np \circ \text{'s mother}) \vdash np\backslash s^\circ}\,/E}{\text{Alice} \circ (\text{saw} \circ (np \circ \text{'s mother})) \vdash s^\circ}\,\backslash E}{\Diamond(\text{Alice} \circ (\text{saw} \circ (np \circ \text{'s mother}))) \vdash \Diamond s^\circ}\,\Diamond I}{\Diamond(\text{Alice} \circ (\text{saw} \circ (np \circ \text{'s mother}))) \vdash s^\circ}\,\text{Unquote}}{\Diamond\text{Alice} \circ (\Diamond\text{saw} \circ (\Diamond np \circ \Diamond\text{'s mother})) \vdash s^\circ}\,\text{K}'\text{ thrice}}{\Diamond\text{Alice} \circ (\Diamond\text{saw} \circ (np \circ \Diamond\text{'s mother})) \vdash s^\circ}\,\text{T}}{(\Diamond\text{Alice} \circ (\Diamond\text{saw} \circ (np \circ \Diamond\text{'s mother}))) \circ_c 1 \vdash s^\circ}\,\text{Root}}{(\Diamond\text{saw} \circ (np \circ \Diamond\text{'s mother})) \circ_c (1 \circ \Diamond\text{Alice}) \vdash s^\circ}\,\text{Right}}{(np \circ \Diamond\text{'s mother}) \circ_c ((1 \circ \Diamond\text{Alice}) \circ \Diamond\text{saw}) \vdash s^\circ}\,\text{Right}}{np \circ_c (\Diamond\text{'s mother} \circ ((1 \circ \Diamond\text{Alice}) \circ \Diamond\text{saw})) \vdash s^\circ}\,\text{Left}}{\Diamond\text{'s mother} \circ ((1 \circ \Diamond\text{Alice}) \circ \Diamond\text{saw}) \vdash np\backslash_c s^\circ}\,\backslash_c I}{\text{a man} \circ (\Diamond\text{'s mother} \circ ((1 \circ \Diamond\text{Alice}) \circ \Diamond\text{saw})) \vdash s^\circ}\,/_c E}{(\text{a man} \circ \Diamond\text{'s mother}) \circ_c ((1 \circ \Diamond\text{Alice}) \circ \Diamond\text{saw}) \vdash s^\circ}\,\text{Left}}{(\Diamond\text{saw} \circ (\text{a man} \circ \Diamond\text{'s mother})) \circ_c (1 \circ \Diamond\text{Alice}) \vdash s^\circ}\,\text{Right}}{(\Diamond\text{Alice} \circ (\Diamond\text{saw} \circ (\text{a man} \circ \Diamond\text{'s mother}))) \circ_c 1 \vdash s^\circ}\,\text{Right}}{\Diamond\text{Alice} \circ (\Diamond\text{saw} \circ (\text{a man} \circ \Diamond\text{'s mother})) \vdash s^\circ}\,\text{Root}}{\text{Alice} \circ (\text{saw} \circ (\text{a man} \circ \text{'s mother})) \vdash s^\circ}\,\text{T thrice}$$

Figure 3: In-situ quantification: deriving *Alice saw a man's mother*

Quantifier	**Type**
a man	$s^\circ /_c (np\backslash_c s^\circ)$
nobody	$s^\circ /_c (np\backslash_c s^-)$
anybody	$s^- /_c (np\backslash_c s^-)$
somebody	$s^+ /_c (np\backslash_c s^+)$
everybody	$s^\circ /_c (np\backslash_c s^+)$

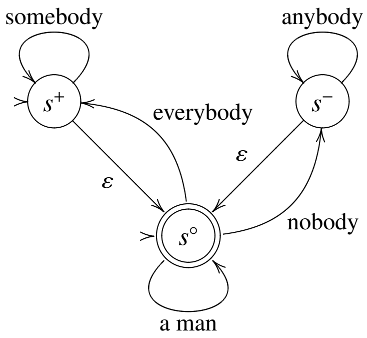


Figure 4: Quantifier type assignments, and a corresponding finite-state machine

3 Polarity sensitivity and evaluation order

The p mode mediates polarity sensitivity. For every unary mode α, we can derive $A \vdash \Box^\downarrow_\alpha \Diamond_\alpha A$ from the rules in Figure 1. This fact is particularly useful when $\alpha = p$, because we assign the types $\Diamond_u \Box^\downarrow_p \Diamond_p s$ and $\Box^\downarrow_p \Diamond_p \Diamond_u s$ to *positive* and *negative* clauses, respectively, and can derive

$$s^\circ \vdash \Diamond_u \Box^\downarrow_p \Diamond_p s, \qquad s^\circ \vdash \Box^\downarrow_p \Diamond_p \Diamond_u s. \tag{9}$$

In words, a neutral clause can be silently converted into a positive or negative one. We henceforth write s^+ and s^- for $\Diamond_u \Box^\downarrow_p \Diamond_p s$ and $\Box^\downarrow_p \Diamond_p \Diamond_u s$. By (9), both types are "subtypes" of s° (that is to say, entailed by s°).

The p mode is used in Figure 5 on the next page to derive *Nobody saw anybody*. Unlike *a man*, the quantifier *anybody* has the type $s^- /_c (np\backslash_c s^-)$, showing that it takes scope over a negative clause to make another negative clause. Meanwhile, the quantifier *nobody* has the type $s^\circ /_c (np\backslash_c s^-)$, showing that it takes scope over a negative clause to make a neutral clause. Thus *nobody* can take scope over the negative clause returned by *anybody* to make a neutral clause, which is complete.

The contrast between (2a) and (3) boils down to the Right (but not Left) postulate's requirement that the leftmost constituent be of the form $\Diamond B$. (In programming-language terms, a subexpression can be evaluated only if all other subexpressions to its left are pure.) For *nobody* to take scope over (and license) *anybody* in (3) requires the context **Anybody saw* _. In other words, the sequent

$$np \circ_c ((1 \circ \Diamond\text{anybody}) \circ \Diamond\text{saw}) \vdash s^- \tag{10}$$

must be derived, in which the Right rule forces the constituents *anybody* and *saw* to be embedded under diamonds. Figure 6 shows an attempt at deriving (10), which fails because the type s^- for negative clauses cannot be Unquoted (shown with question marks). The sequent in (10) cannot be derived, and the sentence **Anybody saw nobody* is not admitted. Nevertheless, *Somebody saw everybody* is correctly predicted to have ambiguous scope, because neutral and positive clauses can be Unquoted.

The quantifiers *a man*, *nobody*, and *anybody* in Figures 3 and 5 exemplify a general pattern of analysis: every polarity-sensitive item, be it traditionally considered a licensor or a licensee, specifies in its type an *input* polarity (of the clause it takes scope over) and an *output* polarity (of the clause it produces). Figure 4 lists more quantifiers and their input and output polarities. As shown there, these type assignments can be visualized as a finite-state machine. The states are the three clause types. The ε-transitions are the two derivability relations in (9). The non-ε transitions are the quantifiers. The start states are the clausal types that can be Unquoted. The final state is the clausal type returned by verbs, namely neutral.

The precise pattern of predictions made by this theory can be stated in two parts. First, due to the lexical types in Figure 4 and the "subtyping" relations in (9), the quantifiers in a sentence must form a valid transition sequence, from widest to narrowest scope. This constraint is standard in type-logical accounts of polarity sensitivity. Second, thanks to the unary modes in the structural

$$\dfrac{\dfrac{\text{nobody}\vdash s^{\circ}/_c(np\backslash_c s^{-})\quad \dfrac{\dfrac{\dfrac{\dfrac{\text{anybody}\vdash s^{-}/_c(np\backslash_c s^{-})\quad \dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\\ \Diamond np\circ(\Diamond\text{saw}\circ np)\vdash s^{\circ}\end{array}}{np\circ_c((1\circ\Diamond np)\circ\Diamond\text{saw})\vdash s^{\circ}}\text{Root, Right, Right}}{\Diamond_p(np\circ_c((1\circ\Diamond np)\circ\Diamond\text{saw}))\vdash\Diamond_p s^{\circ}}\Diamond_p\text{I}}{np\circ_c((1\circ\Diamond np)\circ\Diamond\text{saw})\vdash s^{-}}\Box_p^{\downarrow}\text{I}}{(1\circ\Diamond np)\circ\Diamond\text{saw}\vdash np\backslash_c s^{-}}\backslash_c\text{I}}{\text{anybody}\circ_c((1\circ\Diamond np)\circ\Diamond\text{saw})\vdash s^{-}}/_c\text{E}}{\Diamond np\circ_c((\Diamond\text{saw}\circ\text{anybody})\circ 1)\vdash s^{-}}\text{Right, Right, Left}}{np\circ_c((\text{saw}\circ\text{anybody})\circ 1)\vdash s^{-}}\text{T twice}}{(\text{saw}\circ\text{anybody})\circ 1\vdash np\backslash_c s^{-}}\backslash_c\text{I}}{\text{nobody}\circ_c((\text{saw}\circ\text{anybody})\circ 1)\vdash s^{\circ}}/_c\text{E}}{\text{nobody}\circ(\text{saw}\circ\text{anybody})\vdash s^{\circ}}\text{Left, Root}$$

Figure 5: Polarity licensing: deriving *Nobody saw anybody*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}\dfrac{\begin{array}{c}\vdots\\ \text{anybody}\circ(\text{saw}\circ np)\vdash s^{-}\end{array}}{\Diamond(\text{anybody}\circ(\text{saw}\circ np))\vdash\Diamond s^{-}}\Diamond\text{I}\\ \vdots\,???\\ \Diamond(\text{anybody}\circ(\text{saw}\circ np))\vdash s^{-}\end{array}}{\Diamond\text{anybody}\circ(\Diamond\text{saw}\circ\Diamond np)\vdash s^{-}}\text{K}'\text{ twice}}{\Diamond\text{anybody}\circ(\Diamond\text{saw}\circ np)\vdash s^{-}}\text{T}}{(\Diamond\text{anybody}\circ(\Diamond\text{saw}\circ np))\circ_c 1\vdash s^{-}}\text{Root}}{(\Diamond\text{saw}\circ np)\circ_c(1\circ\Diamond\text{anybody})\vdash s^{-}}\text{Right}}{np\circ_c((1\circ\Diamond\text{anybody})\circ\Diamond\text{saw})\vdash s^{-}}\text{Right}$$

Figure 6: Linear order in polarity licensing: ruling out *Anybody saw nobody* using left-to-right evaluation order

postulates in Figure 2, whenever two quantifiers take inverse rather than linear scope with respect to each other, the transitions must pass through a start state (that is, a clause type that can be Unquoted) in between. This constraint is an empirical advance over previous accounts, which are oblivious to linear order.

The input and output polarities of quantifiers are highly mutually constrained. Take *everybody* for example. If we hold the polarity assignments of the other quantifiers fixed, then the existence of a linear-scope reading for *A man introduced everybody to somebody* forces *everybody* to be input-positive and output-neutral. But then our account predicts that *Nobody introduced everybody to somebody* has a linear-scope reading, unlike the simpler sentence *Nobody introduced Alice to somebody*. This prediction is borne out, as observed by Kroch (1974, pages 121–122) and discussed by Szabolcsi (2004).

Acknowledgments

Thanks to Chris Barker, Raffaella Bernardi, William Ladusaw, Richard Moot, Chris Potts, Stuart Shieber, Dylan Thurston, and three anonymous referees. This work is supported by the United States National Science Foundation Grant BCS-0236592.

References

Raffaella Bernardi and Richard Moot. 2001. Generalized quantifiers in declarative and interrogative sentences. *Journal of Language and Computation*, 1(3):1–19.

Raffaella Bernardi. 2002. *Reasoning with Polarity in Categorial Type Logic*. Ph.D. thesis, Utrecht Institute of Linguistics (OTS), Utrecht University.

Daniel Büring. 2001. A situation semantics for binding out of DP. In Rachel Hastings, Brendan Jackson, and Zsofia Zvolensky, editors, *SALT XI: Semantics and Linguistic Theory*, pages 56–75, Ithaca. Cornell University Press.

Olivier Danvy and Andrzej Filinski. 1990. Abstracting control. In *Proceedings of the 1990 ACM Conference on Lisp and Functional Programming*, pages 151–160, New York, March. ACM Press.

David R. Dowty. 1994. The role of negative polarity and concord marking in natural language reasoning. In Mandy Harvey and Lynn Santelmann, editors, *SALT IV: Semantics and Linguistic Theory*, Ithaca. Cornell University Press.

Matthias Felleisen. 1988. The theory and practice of first-class prompts. In *POPL '88: Conference Record of the Annual ACM Symposium on Principles of Programming Languages*, pages 180–190, New York. ACM Press.

John Fry. 1997. Negative polarity licensing at the syntax-semantics interface. In Philip R. Cohen and Wolfgang Wahlster, editors, *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 144–150, San Francisco. Morgan Kaufmann.

John Fry. 1999. Proof nets and negative polarity licensing. In Mary Dalrymple, editor, *Semantics and Syntax in Lexical Functional Grammar: The Resource Logic Approach*, chapter 3, pages 91–116. MIT Press, Cambridge.

Manfred Krifka. 1995. The semantics and pragmatics of polarity items. *Linguistic Analysis*, 25:209–257.

Anthony S. Kroch. 1974. *The Semantics of Scope in English*. Ph.D. thesis, Massachusetts Institute of Technology. Reprinted by New York: Garland, 1979.

William A. Ladusaw. 1979. *Polarity Sensitivity as Inherent Scope Relations*. Ph.D. thesis, Department of Linguistics, University of Massachusetts, August. Reprinted by New York: Garland, 1980.

Michael Moortgat. 1996a. Categorial type logics. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, chapter 2. Elsevier Science, Amsterdam.

Michael Moortgat. 1996b. Generalized quantification and discontinuous type constructors. In Harry C. Bunt and Arthur van Horck, editors, *Discontinuous Constituency*, pages 181–207. de Gruyter, Berlin.

Richard Moot. 2002. *Proof Nets for Linguistic Analysis*. Ph.D. thesis, Utrecht Institute of Linguistics (OTS), Utrecht University.

Anna Szabolcsi. 2004. Positive polarity—negative polarity. *Natural Language and Linguistic Theory*, 22(2):409–452, May.

Edwin Williams. 1997. Blocking and anaphora. *Linguistic Inquiry*, 28(4):577–628.

Continuations in Natural Language

[Extended Abstract]

Chris Barker
University of California, San Diego
9500 Gillman Drive
La Jolla, CA, USA
barker@ucsd.edu

ABSTRACT

Computer scientists, logicians and functional programmers have studied continuations in laboratory settings for years. As a result of that work, continuations are now accepted as an indispensable tool for reasoning about control, order of evaluation, classical versus intuitionistic proof, and more. But all of the applications just mentioned concern *artificial* languages; what about *natural* languages, the languages spoken by humans in their daily life? Do natural languages get by without any of the marvelous control operators provided by continuations, or can we find continuations in the wild? This paper argues yes: that an adequate and complete analysis of natural language must recognize and rely on continuations. In support of this claim, I identify four independent linguistic phenomena for which a simple CPS-based description provides an insightful analysis.

1. INTRODUCTION

The applications of continuations to date are remarkably diverse. As representative examples, I will mention three strands of research here: first, continuations provide an order-independent way of describing evaluation order in formal languages. For instance, Plotkin [10] shows how various Continuation-Passing Style (CPS) transforms can model evaluation disciplines such as call-by-name or call-by-value. Second, Griffin [5], Murthy [9] and others show that continuations are intimately involved in characterizing the computational content of classical (as opposed to intuitionistic) proofs. Third, continuations provide a useful tool for programmers who want to use powerful control operators such as `call/cc` in Scheme (and its analogs in other functional programming languages); conversely, Queinnec [11] shows how to use continuations in order to write in a direct style and still be assured that programs will behave in a reasonable way when it is the program's *users* who have access to powerful control operators (such as the 'back' button on a web browser).

As diverse as these applications are, they all involve the design and analysis of artificial languages (i.e., programming languages and logical languages), as opposed to natural languages (the sort of languages that humans characteristically use when communicating with each other). We might wonder, then, whether continuations are relevant for the study of natural languages. This is the master question addressed by this paper:

- [**Relevance**] Are there phenomena in natural language that can profitably be analyzed using continuations?

Obviously, I believe that the answer is "yes"! I will try to persuade you to believe likewise by presenting four case studies, each of which has an analysis in terms of continuations.

An affirmative answer to the master question of relevance leads to a number of subquestions, including the following.

- [**Universality**] Even if some natural languages make use of continuation, do all languages make use of continuations?

Languages differ in many ways. For instance, all languages make a distinction between nouns and verbs, but not all languages make a distinction between singular nouns and plural nouns. I have drawn the case studies below exclusively from English, since that simplifies the exposition, but we must remain alert to the possibility that English may be unusual or even unique in the relevant respects. In fact, however, all of the phenomena discussed below have close analogs in many other languages. Coordination in particular is widespread: as far as I know, every language provides some way of saying something equivalent to *John saw Mary and Tom*, where the coordinating conjunction *and* combines two noun phrases (*Mary* and *Tom*) into a single complex noun phrase *Mary and Tom.* If an adequate analysis of coordination depends on continuations, as I argue below, that would strongly suggest that every natural language makes essential use of continuations.

- [**Distribution**] Which specific control operators do natural languages make use of? Which control operators are more 'natural'?

A variety of control operators that make use of continuations have been proposed over the years, some of which differ in fairly subtle ways: `call/cc` versus control ($\mathcal{C}$), **`shift`** and `reset` versus `fcontrol` and `run`, and so on. Sometimes one

Fourth ACM-SIGPLAN Continuation Workshop '04 Venice, Italy

operator can be expressed in terms of another; nevertheless, if one or the other version is more prevalent in natural language, that might suggest that some operators are more 'natural' than others. In the discussion below, I will argue that $\mathcal{C}$ and `fcontrol/run` are remarkably well-suited to handling quantification and focus.

- [**Delimitation**] Does natural language prefer delimited or undelimited continuations?

Historically, the first continuation operators were undelimited (e.g., `call/cc` or `J`). Felleisen [4, 3] proposed delimited continuations (sometimes called 'composable' continuations) such as control ('$\mathcal{C}$') and prompt ('%'). Interestingly, the natural-language phenomena discussed in this paper all seem to require some form of delimitation.

2. PRELIMINARIES

The intrepid reader is welcome to skip directly to the case studies, backtracking to this section only if some assumption or technique seems puzzling.

Theoretical computer scientists think deeply about the nature of formal languages. Many of the tools and techniques that are relevant for the study and the design of formal languages apply also to the study of natural language, including lexical analysis, parsing, and denotational semantics. As a result, computer scientists have highly developed intuitions about languages and about good ways to analyze them. These intuitions will go a long way towards understanding the issues and phenomena discussed below; after all, one of my main goals in this paper is to emphasize similarities between formal languages and natural languages.

Nevertheless, there are significant differences between computer science approaches and linguistic approaches. I have made some effort to present a discussion of natural language in a way that makes sense to a computer scientist—but only up to a point. To the extent that the assumptions and techniques agree with standard practice in computer science, well and good; but when there are differences, I will depend on the reader to trust that the linguistic assumptions are coherent and well motivated, and to keep an open mind about what is the "right" way to model language.

2.1 Declarative sentences denote truth values

As a starting point, it is necessary to think of natural language utterances as expressing computations. Natural language is clearly capable of specifying algorithms: just think of a recipe in a cookbook. Natural language can also express imperatives (*Shut the door!*), queries, and can even impinge on the real world in a direct way that formal languages typically do not (*I hereby declare you husband and wife*).

In this paper, however, I will concentrate exclusively on relatively simple and mundane declarative sentences such as *Everyone left.* Declarative sentences will have a type equivalent to a boolean, and their values will be either `true` or `false`. To see how this makes sense, it may help to consider the role of *everyone left* in the conditional sentence *If (everyone left), then shut the door.*

A little thought will reveal that treating declarative sentences as denoting truth values is inadequate in general. After all, there are more than two possible meanings for sentences. Even if *Everyone left* and *The room is empty* are both true, they have different meanings. But the same objection applies for formal languages: '`x == x`' means something very different from '`x == 3`', yet it still makes sense to treat them both as boolean expressions.

In some more elaborate linguistic treatments, sentences denote functions from times and worlds to truth values, with an analogous shift for expressions of other types. In the parlance of linguistics, a treatment in terms of truth values is 'extensional', and a system with times and worlds is 'intensional'. Shan [13] shows that intensionalization can be rendered as a monad. Intensionality is not crucial in any of the discussions below, and the types will be complex enough anyway, so I will use an extensional semantics on which sentences denote truth values.

2.2 Methodology: formal grammar fragments

It is standard in mainstream linguistics when analyzing a particular type of expression to provide a formal grammar approximating the syntax and the semantics of the expressions under study, and that is the approach that I will take in this paper. Although I intend the grammar fragment developed below to capture something genuine about the nature of the natural language expressions, it is important to bear in mind that any formal grammar is at best an approximation of natural language, i.e., a hypothesis, not a definition, and the degree to which the formal treatment accurately reflects the behavior of the natural language expression will always be an empirical issue.

In the fragment, each expression will be assigned membership in some syntactic category, and the name of each syntactic category will also serve to indicate the semantic type of the meanings of expressions in that category.

A concrete example will help make this clear. Here is a direct-style starting point that the fragment below will build on:

(1) Some direct-style lexical entries:

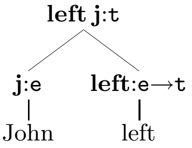
Expression	Syntactic category (= type)	Semantic value
John	`e`	**j**
Mary	`e`	**m**
left	`e→t`	**left**
saw	`e→(e→t)`	**saw**

Intuitively, `e` is the category of expressions that denote individuals. Since the semantic job of the proper name *John* is to refer to a particular individual (in this case, the individual **j**), the word *John* is a member of the category `e`. Expressions in a category with a complex label of the form A→B will denote a function from meanings of type A to meanings of type B. Expressions in category `t` denote truth values (booleans), so that an intransitive verb such as *left* has type `e→t` and denotes a function from individuals to truth values.

Syntactic combination in this direct grammar will always correspond semantically to functional application.

(2) fx:B ::= x:A f:A→B

This way of specifying syntactic combination is a kind of BNF notation enhanced in two ways: categories have been labeled with semantic values, and the syntactic categories contain variables over types. The idea is that (2) schematizes over a set of valid BNF rules for any choice of categories A and B. For instance, (2) licenses combining the noun phrase *John* with the intransitive verb phrase *left* (in that order) by virtue of the following instantiations: $\mathrm{A} = \mathtt{e}$, $\mathrm{B} = \mathtt{t}$, $f = \textbf{left}$, and $x = \mathbf{j}$. This gives rise to the following syntactic and semantic analysis for the declarative sentence *John left.*



The intended interpretation is that a use of the sentence *John left* will be predicted to be true just in case the function denoted by *left* maps the individual referred to by *John* onto the truth value `true`.

Unlike most formal languages, which tend to prefer a uniform direction of functional application, English allows some arguments to precede the function that applies to them, as in the example immediately above, and some to follow. For instance, in the sentence *John saw Mary*, the transitive verb *saw* combines first with *Mary* on its right, then the verb phrase *saw Mary* combines with the argument *John* to its left. Thus we need a second rule for syntactic combination to allow for arguments that appear on the right:

(3) $fx{:}\mathrm{B} \quad ::= \quad f{:}\mathrm{A}\rightarrow\mathrm{B} \quad x{:}\mathrm{A}$

One of the ways in which the formal grammar developed here is unrealistic is that the syntactic categories of *left* and *saw* as given in (1) do not specify which arguments follow and which precede. Thus in addition to deriving *John left* and *John saw Mary*, it also incorrectly derives the ungrammatical sentences **Left John* and **Saw John Mary* (the star indicates that the string it is prefixed to is not well-formed).

It is not difficult to build grammars that take linear order into account; but since order will not play an important role in what follows, I have opted to ignore linear order in an effort to keep the correspondence between syntactic category and semantic type as transparent as possible: since the syntactic category of *saw* is `e→(e→t)`, it is clear that the semantic type of its value will be a function from individuals to functions from individuals to truth values.

2.3 Strategy: case studies

My strategy will be to discuss a number of cases in which a continuation-based analysis is elegant and potentially insightful. I will discuss several different cases rather than developing a single analysis in more depth for two reasons: first, because which solutions strike a responsive cord varies from one person to another; and second, for the sheer joy of considering a number of different natural language phenomena.

There will be four case studies: quantification, coordination, focus particles, and misplaced modifiers. (I will introduce each one of these phenomena using concrete examples below, of course.) The treatments of quantification and coordination have been developed in some detail in [1], but the proposals for focus particles and misplaced modifiers are new to this paper. There are a few other continuation-based analyses of natural language already in the literature: Shan [15] proposes a continuation-based analysis of question formation, and Shan and Barker [17] discuss a continuation approach to the phenomena of weak crossover and superiority, which unfortunately requires background discussions that are too complex to attempt here. In addition, at this conference, Shan will present a new continuation-based analysis of negative polarity.

3. CASE STUDY: QUANTIFICATION

The most carefully worked out natural-language application of continuations to date concerns quantification ([2], [1], [17]). According to the simple analysis above in section 2, the sentence *John left* denotes the truth value (**left j**) and does not involve quantification; a sentence like *Everyone left*, however, is quantificational, and means roughly $\forall x.\textbf{left}\,x$. (We say that the symbol '$\forall$' is a quantifier, and likewise '$\exists$' below.)[1]

The main problem of interest is what I call 'scope displacement': even when a quantificational expression such as *everyone* occurs in a deeply embedded position, the quantifier it contributes can take semantic scope over the entire sentence.

(4) a. John (saw everyone).
  b. $\forall x.\textbf{saw}\,x\,\mathbf{j}$

Here, even though *everyone* is buried within the verb phrase *saw everyone*,[2] in the paraphrase in (4b), the logical quantifier $\forall$ takes scope over the entire meaning. Continuations, of course, are superb at allowing a deeply embedded operator to take control over a larger expression in which it is embedded, and the analysis described here is based on that ability.

3.1 From quantification to continuization

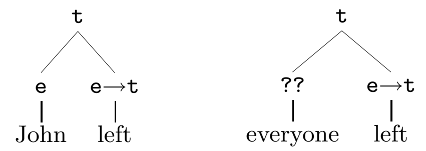
We proceed by deducing a type for quantificational expressions like *everyone*, *no one*, or *someone.* Syntactically, these quantificational expressions can be substituted in any position occupied by a proper name. Thus since *John saw Mary* is grammatical, the sentences *Everyone saw Mary*, *John saw everyone*, *Someone saw everyone*, etc., will all be grammatical as well. Yet it is problematic assigning *everyone* type `e`, the same type as *John* or *Mary*, since there is no particular individual who has all and only the properties that are true of everyone. The inadequacy of supposing that quantificational noun phrases denote individuals is even more stark in the case of *no one*: if *No one left* is true, which individual has the property of leaving?

We can solve this dilemma by considering the parallelism in the following two syntactic analyses:

[1] I use expressions in the first-order predicate calculus to specify what I have in mind for the meaning. At one level, this suggests that natural language can be translated into some suitable logical language, and this is in fact a perfectly legitimate technique. At a deeper level of analysis, however, I am assuming for the purposes of this paper that words are logical constants denoting some specific individual, truth value, or function, and that the denotations of complex phrases are composed from the denotations of their constituent words entirely through functional application as governed by the various syntactic combination rules as given. In Montague's [8] terms, the intermediate logical description language is non-essential, and could be dispensed with entirely if desired.

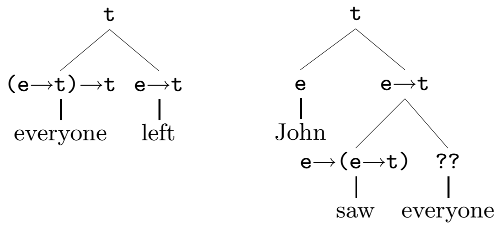
[2] The parentheses in (4a) indicate what I take to be a constituent. There is abundant evidence that the transitive verb *saw* forms a constituent with its direct object *everyone* to the exclusion of the subject *John.* To give just one argument, note that verb phrases can be coordinated (see section 5): *John (saw everyone) and (called home).*

(5)



If *John* has a type that combines with the type of *left* to form a complex expression of type t, then presumably *everyone* must also have a type that combines with the type of *left* to produce an expression of type t. Since we want to avoid giving *everyone* the type e, the next simplest type that will serve is (e→t)→t. This is in fact the main insight proposed by Montague [8]: that noun phrases such as *everyone* and *no one* can have the type (e→t)→t, which he called a *generalized quantifier* (for reasons I won't discuss here). Intuitively, what this analysis says is that *everyone* is a function mapping a verb phrase to a truth value.

(6)



This naive solution works out fine for the example *Everyone left* (although it has the rather disturbing effect of reversing the direction of functional application, a point I'll return to shortly). But if we attempt to place a quantificational expression in some other position where a proper name can occur (i.e., instead of *John saw Mary* we have *John saw everyone*, as in the diagram on the right of (6)), the solution breaks down: the generalized quantifier type (e→t)→t does not suffice, and we seem to need a yet higher type.

This is the point at which continuations enter the picture. Consider again the diagrams in (5) from the point of view of continuations. What is the continuation of the expression *John* relative to the sentence *John left*? It will be a function mapping individuals into truth values, i.e., a function of type e→t. Not coincidentally, this happens to also be the type of the verb phrase *left*. From this perspective, the proposed generalized quantifier type for *everyone*, (e→t)→t, is a function from its continuation (type e→t) to a truth value.

Now we can understand better how to deal with the situation depicted in (6). What is the continuation of *everyone* in the sentence *John saw everyone*? Well, once again it will be a function mapping individuals to truth values. More specifically, it will be the function mapping each individual x to true just in case John saw x. If only we had a way of providing the denotation of *everyone* with access to its continuation, a single type would serve in all of the examples considered so far.

In operational terms, we need the control operator $\mathcal{C}$:

(7) $E[\mathcal{C}M] \triangleright M(\lambda x.E[x])$

What (7) says is that when $\mathcal{C}M$ occurs in an evaluation context E, evaluation proceeds by packaging the context as a continuation (i.e., $\lambda x.E[x]$), then replacing the entire computation with M applied to the continuation.

Then if *everyone* denotes $\mathcal{C}(\lambda P.\forall x.Px)$, *John saw everyone* will denote $(\lambda P.\forall x.Px)(\lambda x.\mathbf{saw}\,x\,\mathbf{j})$, which is equivalent via β-reduction to $\forall x.\mathbf{saw}\,x\,\mathbf{j}$, as desired.

My denotational implementation of this analysis will be based on the following CPS transform:

(8) a. $\overline{\alpha} = \lambda\kappa.\kappa\alpha$ (for any constant α)
b. $\overline{MN} = \lambda\kappa.\overline{M}(\lambda m.\overline{N}(\lambda n.\kappa(mn)))$

I prove a simulation result for this transform in [1].

Note that the transform does not provide a rule for expressions of the form $\lambda x.M$. It is possible to omit such a rule here because the natural language constructions considered in this paper involve only functional application, and never lambda abstraction.[3]

As a consequence, the type of the CPS transform of an expression of any direct type X will be $(\mathrm{X}\to\sigma)\to\sigma$, where σ is some answer type. In particular, the type of the CPS transform of a direct expression of type A→B will simply be $((\mathrm{A}\to\mathrm{B})\to\sigma)\to\sigma$, rather than the more traditional (e.g., [7]) type $\mathrm{A}'\to((\mathrm{B}'\to\sigma)\to\sigma)$, where X′ is the type of the transform of an expression of type X. This difference in types accounts for the expression '$\kappa(mn)$' in (8b) rather than the traditional '$mn\kappa$'. I find this typing much simpler; however, it seems to be wrenching for people used to the standard transform, so this is one of the places where I must ask the reader to keep an open mind about the right way to do things.

The CPS transform of the lexical items in (1) all depend on the rule governing constants in (8a):

(9) CPS transforms of the lexical entries in (1):

John	(e→A)→A	$\lambda\kappa.\kappa\,\mathbf{j}$
Mary	(e→A)→A	$\lambda\kappa.\kappa\,\mathbf{m}$
left	((e→t)→A)→A	$\lambda\kappa.\kappa\,\mathbf{left}$
saw	((e→(e→t))→A)→A	$\lambda\kappa.\kappa\,\mathbf{saw}$

Thus for the purposes of the transform, *left* counts as a constant, even though its direct denotation **left** is a function of type e→t.

Here A is a variable over syntactic categories (in effect, over types). For the sake of quantification alone, we could replace the type variables here with t. The more general rules will be needed, however, when we extend the fragment to handle focus in the next section.

The syntactic combination rules for the CPS grammar will be as follows:

(10) $\lambda\kappa.\overline{\mathrm{M}}(\lambda m.\overline{\mathrm{N}}(\lambda n.\kappa(mn))){:}(\mathrm{B}\to\mathrm{E})\to\mathrm{D}$
$::= \quad \overline{\mathrm{M}}{:}((\mathrm{A}\to\mathrm{B})\to\mathrm{C})\to\mathrm{D} \qquad \overline{\mathrm{N}}{:}(\mathrm{A}\to\mathrm{E})\to\mathrm{C}$

(11) $\lambda\kappa.\overline{\mathrm{N}}(\lambda n.\overline{\mathrm{M}}(\lambda m.\kappa(mn))){:}(\mathrm{B}\to\mathrm{C})\to\mathrm{E}$
$::= \quad \overline{\mathrm{N}}{:}(\mathrm{A}\to\mathrm{D})\to\mathrm{E} \qquad \overline{\mathrm{M}}{:}((\mathrm{A}\to\mathrm{B})\to\mathrm{C})\to\mathrm{D}$

These syntactic combination rules just make explicit how to syntactically and semantically combine expressions that have already undergone the CPS transform. Thus (10) is the CPS analog of (3), and (11) is the analog of (2).

Some of the complexity in (10) and (11) could be reduced if we chose C = D = E. This would suffice for present purposes except for the treatment of focus in the next section, which depends on allowing a control operator to return an answer that is not of type t.

[3] It is an interesting question whether this abstraction-free approach can be maintained in a more complete analysis of natural language. One strategy is to use continuations instead of lambda abstraction whenever natural language seems to require lambda abstraction; see, e.g., [17] for a continuation-based treatment of relative clause formation.

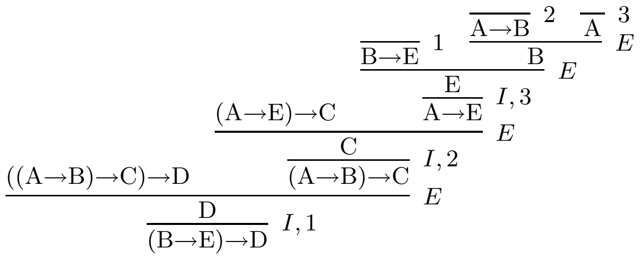
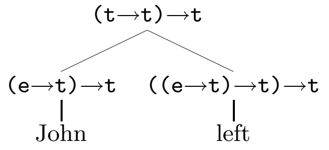


Figure 1: Derivation of (10): a natural-deduction proof using only → Elimination and → Introduction that ((A→B)→C)→D, (A→E)→C ⊢ (B→E)→D.

The proof in figure 1 may help untangle the types. The semantic values in (10) (and hence the application rule in the CPS transform given in (8b)) are just the Curry-Howard labelling of the proof in figure 1. An analogous derivation is available for (11), but is not included here.

These rules give us the following derivation:



$$
\begin{aligned}
[\![\textit{John left}]\!] &= \lambda\kappa.[\lambda\kappa.\kappa\textbf{left}](\lambda f.[\lambda\kappa.\kappa\textbf{j}](\lambda x.\kappa(f x))) \\
&\rightsquigarrow \kappa.\kappa(\textbf{left}\,\textbf{j})
\end{aligned}
$$

which, after application to the trivial continuation $\lambda x.x$, yields the same denotation provided by the direct grammar, namely, **left j**.

At this point we can add lexical entries for some quantificational noun phrases that have no direct counterpart.

(12)	everyone	`(e→t)→t`	$\lambda\kappa.\forall x.\kappa x$
	someone	`(e→t)→t`	$\lambda\kappa.\exists x.\kappa x$

Since *everyone* has the same type as the CPS transform of *John*, we now have

$$
\begin{aligned}
[\![\textit{Everyone left}]\!] &= \lambda\kappa.[\lambda\kappa.\kappa\textbf{left}](\lambda f.[\lambda\kappa.\forall x.\kappa x](\lambda x.\kappa(f x))) \\
&\rightsquigarrow \lambda\kappa.\forall x.\kappa(\textbf{left}\,x),
\end{aligned}
$$

as well as

$$[\![\textit{John saw everyone}]\!] \rightsquigarrow \lambda\kappa.\forall x.\kappa(\textbf{saw}\,x\,\textbf{j})$$

which yield $\forall x.\textbf{left}\,x$ and $\forall x.\textbf{saw}\,x\,\textbf{j}$ after application to the trivial continuation, as desired.

Note that continuization has restored the natural direction of functional application (which had been reversed by the first strategy depicted in (6)). That is, not only is the direction of (the CPS analog of) functional application in *John left* right-to-left, the direction of application in the continuized grammar for handling *Everyone left* is also right-to-left.

3.2 Quantificational ambiguity

The transform as specified in (8b) imposes a left-to-right evaluation discipline. But we might just as well have used instead

(13) $\overline{MN} = \lambda\kappa.\overline{N}(\lambda n.\overline{M}(\lambda m.\kappa(mn)))$

in which the right-hand component, $\overline{\text{N}}$, is evaluated first. For sentences containing only one quantificational noun phrase, order of evaluation makes no difference. But for sentences containing two or more quantificational noun phrases, a difference in order of evaluation corresponds to a difference in the predicted meaning:

(14) a. Someone saw everyone.
b. $\exists x\forall y.\textbf{saw}\,x\,y$
c. $\forall y\exists x.\textbf{saw}\,x\,y$

Given the availability of both evaluation orders, the sentence in (14a) containing two quantificational noun phrases will have meanings that (after application to the trivial continuation) reduce to either (14b) (for left-to-right evaluation) or (14c) (for right-to-left evaluation).

The consensus in the field is that sentences like (14a) are in fact ambiguous in exactly the way considered here. The interpretation in (14b) corresponds to a situation in which there is some particular person who has the property of seeing everyone, and the interpretation in (14c) corresponds to a situation in which there is a potentially different see-er who saw each person.

Quantifier scope ambiguity has been studied in considerable detail. For more discussion of the predictions of continuation-based analyses with respect to this type of ambiguity, see [1], [2] and [17].

4. CASE STUDY: FOCUS

In this section I propose that a phenomenon called ‘focus’, along with so-called focus particles such as *only*, behave like Sitaram’s [18, 19] `run` and `fcontrol`. One point of interest is that these control operators rely on delimited continuations. Delimited continuations have received a considerable amount of attention recently, and in fact are the topic of at least three papers at the workshop. I believe that they play an important role in natural language as well. In fact, I would be (mildly) surprised to find a natural language operator that behaved exclusively in an undelimited manner.

4.1 `fcontrol` and `run`

There are many control operators that rely on delimited continuations, including Felleisen’s control and prompt, shift and reset, and others. I will discuss Sitaram’s `fcontrol` and `run` here for the simple reason that they provide exactly the functionality required for describing the behavior of focus particles such as *only*.

Roughly, `fcontrol` provides a way to throw a signal, and `run` catches and handles signals thrown by `fcontrol`.

The `fcontrol` operator takes one argument, and throws two values: the argument to `fcontrol` and the prefix of the continuation of the `fcontrol` expression up to the closest enclosing `run`.

The `run` operator takes two arguments: an expression possibly containing an occurrence of `fcontrol`, and a “handler” function that processes the two arguments thrown by `fcontrol`.

For instance, in

```
(+ 1 (run (* 2 (+ (fcontrol 3) 4))
          (lambda (x k) (k (k x)))))
```

The invocation of `fcontrol` throws the two values 3 and a (truncated) continuation κ. Since κ is the continuation of the expression “`(fcontrol 3)`” relative to the larger expression only up to the closest occurrence of `run`, κ will be the function $\lambda y.(*\,2\,(+\,y\,4))$. The handler procedure will bind `x` to 3 and `k` to κ, and the result of the entire computation will be 37.

The continuation caught by the `fcontrol` is delimited, in the sense that it does not contain any material outside the scope of the enclosing `run`; in particular, the function corresponding to the continuation does not contain the increment operation (there is no "+ 1" inside of κ). In this example, since the continuation is just a function, it can be called several times without replacing the current evaluation context. This allows expressions like `(k (k x))` as in the example at hand, in which one occurrence of a continuation takes an expression involving another occurrence of the same continuation as an argument. This is the sense in which delimited continuations are 'composable'.

Before we can make use of `run` and `fcontrol`, we must consider the meaning of sentences containing focus marking and focus particles.

4.2 Focus

Most (probably all) languages provide some way of marking some constituent in a sentence as having extra prominence. In spoken English, this is typically accomplished in part by a local maximum in the fundamental frequency (the lowest frequency at which the vocal folds are vibrating). By convention, the location of such a 'pitch accent' is indicated typographically by setting the most affected word in capital letters:

(15) a. JOHN saw Mary.
b. John SAW Mary.
c. John saw MARY.

There is a distinct but elusive difference in meaning among these sentences that depends on the location of the pitch accent. In each case, it remains true that John saw Mary, but which piece of information is being emphasized differs. In traditional terms, the constituent containing the pitch accents is said to be 'in focus', which means (very roughly) that it carries the new information provided by the sentence.

These observations can be sharpened by noting that the location of the pitch accent correlates with the precise piece of information requested by a question.

(16) a. Who saw Mary?
b. What did John do to Mary?
c. Who did John see?

Thus (15a) can be a suitable answer only to the question in (16a), and not to either (16b) or (16c), and similarly for the other answer/question pairs.

The semantic effect of the location of pitch accent becomes even more tangible in the presence of a focus particle such as *only*, *also*, or *too*.

(17) a. John only drinks PERrier.
b. John only DRINKS Perrier.

We say that *only* 'associates' with whatever element is in focus. With a pitch accent on the first syllable of *Perrier*, then, *Perrier* will be in focus, and (17a) conveys at least the information paraphrased in (18):

(18) a. John drinks Perrier.
b. There is nothing else that John drinks
other than Perrier.

The particle *only*, then, picks out some element in the situation and contrasts it with other possible alternative choices: John doesn't drink whiskey, he doesn't drink milk, he only drinks Perrier.

With a pitch accent on the verb *drinks* in (17b), however, the appropriate paraphrases differ:

(19) a. John drinks Perrier.
b. There is nothing else that John does with Perrier
other than drink it.

It remains true that John drinks Perrier, but now the element of the situation that *only* puts into contrast with other alternatives is the activity of drinking. According to (17b), all John does with Perrier is drink it: he doesn't sell it, he doesn't photograph it, he doesn't bathe in it.

Note that the conditions under which (17a) and (17b) will be true are mutually distinct: (17a) can be true even if John sometimes bathes in Perrier, and (17b) can be true even if John sometimes drinks whiskey. Thus in the presence of *only*, the location of the pitch accent can determine whether a sentence is true or false.

I will treat pitch accent in a sentence as if it were an operator F immediately preceding the constituent that is in focus (i.e., that bears the pitch accent). I will continue to use capitals to help guide pronunciation.

(20) a. John only drinks F(PERrier).
b. John only F(DRINKS) Perrier.

Now we're ready to attempt an analysis in terms of `run` and `fcontrol`. The key is to ask the following question: given pitch accent on *Perrier*, what precisely is the relation that holds between John and Perrier and nothing else?

(21) a. John only drinks F(PERrier).
b. $\lambda xy.\mathbf{drink}\,x\,y$

The answer is the relation that holds between John and some object x if John drinks x. In other words, (21b) is a continuation, namely, the continuation of the focussed expression delimited by the enclosing *only*.

Considering next the contrasting example with pitch instead on *drinks*, what is the relation that holds between John and the activity of drinking and nothing else?

(22) a. John only F(DRINKS) Perrier.
b. $\lambda xy.x\ \mathbf{Perrier}\ y$

The answer this time is the relation that holds between John and some activity x if John does x to Perrier. Once again, the desired relation is the continuation of the focussed word up to but not including *only*.

Based on these examples, we can now guess that the semantic effect of pitch accent on a constituent is exactly `fcontrol`.[4] So where I wrote 'F' in (22), the meaning is `fcontrol`. (In a happy coincidence, in this application we can construe the 'f' of `fcontrol` as mnemonic for 'focus'.) Similarly, the meaning of *only* will invoke `run`. It remains only to specify the handler routine that unpacks the information provided by the use of `fcontrol`:

$$[\![\mathit{only}\ P]\!] = \mathtt{run}P\lambda x\kappa y.(\mathbf{and}\ (\kappa xy)\ (\forall z(\mathbf{or}(\mathbf{equal}\,x\,z)(\mathbf{not}(\kappa zy)))))$$

This denotation gives rise to the following analyses:

(23) John only drinks F(PERrier).
(and (drinks Perrier j)
(∀ z **(or (equal Perrier** z)
(not (drinks z **j)))))**

(24) John only F(DRINKS) Perrier.
(and (drinks Perrier j)
(∀ z **(or (equal drinks** z)
(not (z **Perrier j)))))**

[4] Chung-chieh Shan first noticed the similarity between my analysis of focus and Sitaram's operators.

These meanings are equivalent to the paraphrases we started with in (18) and (19).

Now, it only makes sense to bother building continuations if the meaning captured by the continuation can be arbitrarily complex. The examples we have discussed so far have been as simple as possible, so it is worth considering examples in which the delimited continuation is more complicated.

(25) Mary only tried to dance with F(JOHN).
  (**and** (**tried-to-dance-with j m**)
    ($\forall$ z (**or** (**equal j** z)
      (**not** (**tried-to-dance-with** z **m**)))))

(26) Mary only tried to F(DANCE) with John.
  (**and** (**tried-to-dance-with j m**)
    ($\forall$ z (**or** (**equal dance** z)
      (**not** (**tried-to-**z**-with j m**)))))

The simple denotational fragment in section 8 does not attempt to reconstruct the full power of `run` and `fcontrol`, but it does handle the examples discussed here.[5]

5. CASE STUDY: COORDINATION

One of the distinctive features of natural language is the pervasive use of polymorphic coordination.

(27) a. [John left] and [Mary left]. `t`
  b. John [left] and [slept]. `e→t`
  c. John [saw] and [remembered] Mary. `e→(e→t)`
  d. [John] and [Mary] left. `e`

The types in the right column of (27) correspond to the (direct, pre-CPS) type of the bracketed expressions coordinated by *and*. In (27a), two clauses (type `t`) coordinate to form a complex sentence; in (27b), two verb phrases (type `e→t`) coordinate to form a complex verb phrase; and so on.

What about quantificational noun phrases? They also can coordinate:

(28) a. Someone or everyone left.
  b. John or everyone left.

Even more interesting, they can more or less freely coordinate with proper names as in (28b), providing confirmation that proper names and quantificational noun phrases are syntactically interchangeable, as predicted by the analysis in section 3.

We can arrive at a continuation-based analysis of coordination if we consider that (27d) means the same thing as *John left and Mary left.*[6] The continuation of the coordinated phrase is $\lambda x.$**left** x; what the conjunction does is take this continuation and distribute it across each of the conjuncts. The resulting analysis of conjunction involves adding two new rules for syntactic combination:

(29) $\lambda\kappa.$**and**$(\overline{\mathrm{L}}\kappa)(\overline{\mathrm{R}}\kappa)$:A ::= $\overline{\mathrm{L}}$:A "and" $\overline{\mathrm{R}}$:A

(30) $\lambda\kappa.$**or**$(\overline{\mathrm{L}}\kappa)(\overline{\mathrm{R}}\kappa)$:A ::= $\overline{\mathrm{L}}$:A "or" $\overline{\mathrm{R}}$:A

These rules differ from one another only in substitution of *or* for *and*.

The syntactic parts of these rules simply say that anywhere that an expression of type A can occur, an expression of the form "A_1 and A_2" or "A_1 or A_2" can also occur, as long as A_1 and A_2 are themselves expressions of category A.

What the semantic parts of the rules say is: whatever you were planning on doing to the value provided by the expression in this position, first do it to the value of the left conjunct, then do it to the value of the right conjunct, and conjoin the two results. This schema guarantees the following example paraphrases:

(31) a. John left and slept. John left and John slept.
  b. John and Mary left. John left and Mary left.
  c. John or everyone left. John left or everyone left.

One sign of the utility of coordination is that in Wall Street Journal text, *and* is the second most commonly used word (first place goes to *the*). If suitably constrained with syntactic marking (such as parentheses) overtly marking the syntactic strings involved, coordination as a control operator could provide a rather appealing programming device. Imagine being able to write `if (x == (2 or 3)) then ...` and have it mean `if ((x == 2) or (x == 3)) then ...`.[7]

6. CASE STUDY: MISPLACED MODIFIERS

As a final example of a natural language construction that might profit from a continuation-based analysis, consider the following data:

(32) a. An occasional sailor walked by.
  b. John drank a quiet cup of tea.

The modifier *occasional* is misplaced:[8] there is no specific sailor (nor even any set of sailors) that has the property of being occasional; rather, it is the event of a sailor walking by that happens occasionally compared with other relevant events. Similarly, in (32b), it is not the cup of tea that is quiet in the relevant sense, but the activity of drinking the tea.

Once again, we have an embedded element that needs to take semantic force at a higher level, and once again we can allow that expression to take control by providing it with access to its continuation. (Chris Potts (personal communication) first suggested using continuations to analyze misplaced modifiers, though he shouldn't be held responsible for the shortcomings of my treatment here.) Assuming we know what the adverb *occasionally* means, we can give the misplaced adjective *occasional* the following denotation: $\mathcal{C}\lambda\kappa.$**occasionally**$(\kappa\lambda x.x)$, and similarly for *quietly*

[5] There is an important dependence on the context of utterance that goes unrecognized in this analysis. If *John only DRANK Perrier* quantified over absolutely everything that John might have done with Perrier, then it would entail that he didn't buy Perrier, that he didn't swallow Perrier, that he didn't taste Perrier, that he didn't raise Perrier to his lips, that he didn't do all kinds of things to Perrier that he must have done. The standard assumption is that the quantification over alternatives must be context-dependent, so that what a use of *only* really means is that there is no other *contextually-relevant* thing that John did to Perrier. How precisely to calculate what ought to count as a contextually-relevant activity is a problem for AI, and not for linguistics.

[6] There are other kinds of coordination not treated here. For instance, *John and Mary are a happy couple* cannot be paraphrased as **John is a happy couple and Mary is a happy couple.* Interestingly, unlike conjunction, disjunction displays only the kind of behavior characterized by the rule given in (30).

[7] Hayo Thielecke points out that J, an APL-like language whose web site (`http://www.jsoftware.com`) claims that it uses "constructs and syntax which closely mirror those of natural language", has a construction that is a special case of the operator imagined here. More specifically, under certain circumstances `x (f g h) y` evaluates as `(x f y) g (x h y)`, which is isomorphic to example (27c).

[8] If you prefer fancy terminology, this is a type of hypallage.

and *quiet*. The idea is to replace the misplaced adjective with the trivial adjective meaning $\lambda x.x$, and then let its adverbial counterpart take scope over the complete sentence.

This analysis gives the following equivalences:

(33) a. An occasional sailor walked by.
Occasionally, a sailor walked by.

b. John drank a quiet cup of tea.
Quietly, John drank a cup of tea.

Additional details (such as the type of adjectives and nouns) are given in the cumulative fragment below in section 8.

7. HISTORICAL NOTES

If natural language is rife with phenomena that beg for a continuation-based analysis, why hasn't anyone noticed before? The answer, of course, is that some people *have* noticed. In fact, I would suggest that in addition to the many independent discoveries of continuations described by Reynolds [12], Richard Montague [8] also invented a technique that relies in a limited way on continuations.

Montague was a logician at UCLA who was one of the major figures in the early days of establishing formal approaches to natural language semantics. In 1970 he constructed the first popular formal analysis of quantification [8]. He did this by suggesting that quantificational noun phrases such as *everyone* differ from non-quantificational noun phrases such as *John* in just the way I proposed in section 3: quantificational noun phrases are in effect control operators whose meanings are properly expressed as functions on their own continuations.

The main limitation of Montague's approach is that noun phrases were the only type of expression that had access to their continuations; nevertheless, with hindsight, anyone familiar with continuation-passing style programming will immediately recognize a primitive form of continuation passing in Montague's formal grammars. Joe Goguen, my colleague at UCSD, was a colleague of Montague's at UCLA during the 70's, and he tells me that the connection between Montague's techniques and continuations was noticed long ago, at least in the folklore, if not in the literature.

It has only been recently, however, that people have explored this connection systematically, providing a more general mechanism for accessing continuations in natural language analyses. In particular, Herman Hendriks' 1993 dissertation [6] proposes a type-shifting system that in effect produces CPS transforms as needed (and may deserve to be counted as yet another independent invention of a continuation-based system). Philippe de Groote [2] provides an analysis of simple quantification based on the $\lambda\mu$-calculus, and a paper by me [1] explores a continuation-based approach to quantification in considerable detail. Since then, Chung-chieh Shan has proposed a number of continuation-based analyses of natural language phenomena ([14], [15], [16]), including his paper at this conference.

Incidentally, if natural languages do make abundant use of continuations, then Reynolds was doubly right to speak of the 'discovery' of continuations rather than their 'invention'. It is intriguing to consider the possibility that that the presence of continuation-based control operators in the native language of the various discoverers may even have inspired their proposals at some subconscious level.

8. CUMULATIVE FRAGMENT WITH RESET

Figure 2 gives a CPS grammar describing a fragment of English with a context-free syntax and a denotational semantics. The fragment includes versions of the four analyses discussed above for quantification, focus, coordination, and misplaced modifiers.

In addition, the fragment provides a reset operator, associated here with the complementizer *that* (as discussed immediately below). This section will argue for all of the four phenomena that the relevant continuations must at least sometimes be delimited.

The meanings listed in the examples below are guaranteed to be equivalent to the denotations provided by the fragment up to β-reduction and application to the trivial continuation $\lambda x.x$.

For instance, here are some simple examples involving zero, one, and two quantificational noun phrases:

(34) a.	John left.	$\mathbf{left}\,\mathbf{j}$
b.	Everyone left.	$\forall x.\mathbf{left}\,x$
c.	John saw Mary.	$\mathbf{saw}\,\mathbf{m}\,\mathbf{j}$
d.	John saw everyone.	$\forall x.\mathbf{saw}\,x\,\mathbf{j}$
e.	Someone saw everyone.	$\exists x.\forall y.\mathbf{saw}\,y\,x$

Since the fragment does not include extra combination rules for right-to-left evaluation, there is only one interpretation for *Someone saw everyone* (see section 3.2 for discussion).

I now introduce embedded clauses and a reset operator.

(35) a.	John claimed [Mary left].	$\mathbf{claim}\,(\mathbf{left}\,\mathbf{m})\,\mathbf{j}$
b.	John claimed [everyone left].	$\forall x.\mathbf{claimed}\,(\mathbf{left}\,x)\,\mathbf{j}$
c.	John claimed that [everyone left].	$\mathbf{claimed}\,(\forall x.\mathbf{left}\,x)\,\mathbf{j}$

Unlike *saw*, which denotes a relation between two individuals, *claimed* relates an individual and a proposition. Syntactically, *claimed* takes a clause as its first argument. Thus the bracketed strings in (35) are all complete clauses in their own right.

The interpretation in (35b) says that John made several different claims, one for each person. The interpretation in (35c) says that he made one single claim, a claim about everybody. Both interpretations seem to be valid.

In continuation terms, we need to be able to delimit the continuation for *everyone* so as to extend only as far as the material in the embedded clause. In order to experiment with delimitation, I will adopt the following expository strategy: as shown by comparing (35b) with (35c), *claimed* optionally allows its argument clause to be introduced by the complementizer *that*. By giving *that* the semantics of a reset operator, we can add or subtract delimitation at will and see what happens. As shown in (35c), for instance, the presence of the *that* delimits the embedded clause and produces the second legitimate interpretation.

The following set of sentences illustrates the analysis of focus.

left	$((\mathtt{e}\to\mathtt{t})\to\mathrm{A})\to\mathrm{A}$	$\lambda\kappa.\kappa\,\mathbf{left}$
saw	$((\mathtt{e}\to(\mathtt{e}\to\mathtt{t}))\to\mathrm{A})\to\mathrm{A}$	$\lambda\kappa.\kappa\,\mathbf{saw}$
claimed	$((\mathtt{t}\to(\mathtt{e}\to\mathtt{t}))\to\mathrm{A})\to\mathrm{A}$	$\lambda\kappa.\kappa\,\mathbf{claimed}$
John	$(\mathtt{e}\to\mathtt{t})\to\mathtt{t}$	$\lambda\kappa.\kappa\,\mathbf{j}$
Mary	$(\mathtt{e}\to\mathtt{t})\to\mathtt{t}$	$\lambda\kappa.\kappa\,\mathbf{m}$
everyone	$(\mathtt{e}\to\mathtt{t})\to\mathtt{t}$	$\lambda\kappa.\forall x.\kappa x$
someone	$(\mathtt{e}\to\mathtt{t})\to\mathtt{t}$	$\lambda\kappa.\exists x.\kappa x$
a	$(((\mathtt{e}\to\mathtt{t})\to\mathtt{e})\to\mathtt{t})\to\mathtt{t}$	$\lambda\kappa.\kappa\,\mathbf{a}$
sailor	$((\mathtt{e}\to\mathtt{t})\to\mathtt{t})\to\mathtt{t}$	$\lambda\kappa.\kappa\,\mathbf{sailor}$
tall	$(((\mathtt{e}\to\mathtt{t})\to(\mathtt{e}\to\mathtt{t}))\to\mathtt{t})\to\mathtt{t}$	$\lambda\kappa.\kappa\,\mathbf{tall}$
occasional	$(((\mathtt{e}\to\mathtt{t})\to(\mathtt{e}\to\mathtt{t}))\to\mathtt{t})\to\mathtt{t}$	$\lambda\kappa.\mathbf{occasionally}(\kappa(\lambda x.x))$

$\lambda\kappa.\overline{\mathrm{M}}(\lambda m.\overline{\mathrm{N}}(\lambda n.\kappa(mn))):(\mathrm{B}\to\mathrm{E})\to\mathrm{D}$	::=	$\overline{\mathrm{M}}:((\mathrm{A}\to\mathrm{B})\to\mathrm{C})\to\mathrm{D}$	$\overline{\mathrm{N}}:(\mathrm{A}\to\mathrm{E})\to\mathrm{C}$	
$\lambda\kappa.\overline{\mathrm{N}}(\lambda n.\overline{\mathrm{M}}(\lambda m.\kappa(mn))):(\mathrm{B}\to\mathrm{C})\to\mathrm{E}$	::=	$\overline{\mathrm{N}}:(\mathrm{A}\to\mathrm{D})\to\mathrm{E}$	$\overline{\mathrm{M}}:((\mathrm{A}\to\mathrm{B})\to\mathrm{C})\to\mathrm{D}$	
$\lambda\kappa.\mathbf{and}(\overline{\mathrm{L}}\kappa)(\overline{\mathrm{R}}\kappa):\mathrm{A}$	::=	$\overline{\mathrm{L}}:\mathrm{A}$	"and"	$\overline{\mathrm{R}}:\mathrm{A}$
$\lambda\kappa.\mathbf{or}(\overline{\mathrm{L}}\kappa)(\overline{\mathrm{R}}\kappa):\mathrm{A}$	::=	$\overline{\mathrm{L}}:\mathrm{A}$	"or"	$\overline{\mathrm{R}}:\mathrm{A}$
$\lambda\kappa.\langle\overline{\mathrm{X}},\kappa\rangle:\mathrm{A}\to((\mathrm{A}\to\mathtt{t})\ \times\ \mathrm{A})$	::=	"F"	$\overline{\mathrm{X}}:\mathrm{A}\to\mathtt{t}$	
$\lambda\overline{\mathrm{X}}\kappa.\mathbf{only}(\overline{\mathrm{X}}\kappa):\mathrm{A}\to\mathtt{t}$	::=	"only"	$\overline{\mathrm{X}}:\mathrm{A}\to((\mathrm{B}\to\mathtt{t})\ \times\ \mathrm{B})$	
$\lambda\kappa.\kappa(\overline{\mathrm{X}}(\lambda x.x)):\mathrm{A}$	::=	"that"	$\overline{\mathrm{X}}:\mathrm{A}$	

Figure 2: A grammar in continuation passing style covering most of the examples discussed in the text.

(36) a. John only saw F (MARY).
$\mathbf{only}\langle\lambda\kappa(\kappa\mathbf{m}), \lambda x.\mathbf{saw}\,x\,\mathbf{j}\rangle$

b. John only F (SAW) Mary.
$\mathbf{only}\langle\lambda\kappa.\kappa\,\mathbf{saw}, \lambda x.x\,\mathbf{m}\,\mathbf{j}\rangle$

c. John only F (SAW MARY).
$\mathbf{only}\langle\lambda\kappa.\kappa(\mathbf{saw}\,\mathbf{m}), \lambda x.x\mathbf{j}\rangle$

d. John only claimed F (MARY LEFT).
$\mathbf{only}\langle\lambda\kappa.\kappa(\mathbf{left}\,\mathbf{m}), \lambda x.\mathbf{claimed}\,x\,\mathbf{j}\rangle$

e. John claimed Mary only saw F (TOM).
*$\mathbf{only}\langle\kappa.\kappa\mathbf{t}, \lambda x.\mathbf{claimed}\,(\mathbf{saw}\,x\,\mathbf{m})\,\mathbf{j}\rangle$

f. John claimed that Mary only saw F (TOM).
$\mathbf{claimed}\,(\mathbf{only}\langle\kappa.\kappa\mathbf{t}, \lambda x.\mathbf{saw}\,x\,\mathbf{m}\rangle)\,\mathbf{j}$

Here **only** is a function taking an ordered pair $\langle\overline{\mathrm{X}},\kappa\rangle$ and returning `true` just in case $\overline{\mathrm{X}}\kappa$ is true and there is no other (contextually relevant) meaning $\overline{\mathrm{Y}}$ of the same type as $\overline{\mathrm{X}}$ such that $\overline{\mathrm{Y}}\kappa$ is true. Thus the interpretation for *John only saw MARY* given (36a) entails that John saw Mary and that there is no other individual that John saw.

Examples (36a) and (36b) show that the pitch accent marker F can take arguments of different type (here, a transitive verb *saw* rather than a noun phrase *Mary*).

Examples (36b), (36c), and (36d) show that the focus marker can take either a single word (e.g., *SAW*), a complex phrase (*SAW MARY*), or even an entire clause (*MARY LEFT*) as its argument. (In the last case, the only thing that John claimed was the Mary left; he did not claim that Tom left, he did not claim that Mary called, etc.)

Comparing (36e) with (36f), we see that once again embedded clauses must be delimited in order to arrive at the correct interpretation: (36e) does not mean that Tom was the only person that John claimed Mary saw; rather, it means that John claimed that Tom was the only person Mary saw. Thus the presence of the reset operator provided by *that* in (36f) is crucial to correctly delimiting the scope of *only.*

The next pair of examples shows the interaction of focus with quantificational noun phrases.

(37) a. John only F(SAW) someone.
$\mathbf{only}\langle\lambda\kappa.\kappa\,\mathbf{saw}, \lambda y.\exists x.y\,x\,\mathbf{j}\rangle$

b. John only F (SAW SOMEONE).
$\mathbf{only}\langle\lambda\kappa.\exists x.\kappa(\mathbf{saw}\,x), \lambda x.x\,\mathbf{j}\rangle$

c. John only saw F (SOMEone).
$\mathbf{only}\langle\lambda\kappa.\exists x.\kappa\,x, \lambda x.\mathbf{saw}\,x\,\mathbf{j}\rangle$

Because the focus marker F takes a continuized meaning as its argument, it is able to handle quantificational foci with no trouble.

The next examples demonstrate the analysis of coordination.

(38) a. John left and Mary left. $\mathbf{and}(\mathbf{left}\,\mathbf{j})(\mathbf{left}\,\mathbf{m})$

b. John and Mary left. $\mathbf{and}(\mathbf{left}\,\mathbf{j})(\mathbf{left}\,\mathbf{m})$

c. John saw Mary and left. $\mathbf{and}(\mathbf{saw}\,\mathbf{m}\,\mathbf{j})(\mathbf{left}\,\mathbf{j})$

d. John saw Mary or everyone. $\mathbf{or}(\mathbf{saw}\,\mathbf{m}\,\mathbf{j})(\forall x.\mathbf{saw}\,x\,\mathbf{j})$

Note that (38d) involves coordinating a proper name with a quantificational noun phrase, which works fine.

(39) a. John claimed Mary or Tom left.
$\mathbf{or}(\mathbf{claimed}\,(\mathbf{left}\,\mathbf{m})\,\mathbf{j})(\mathbf{claimed}\,(\mathbf{left}\,\mathbf{t})\,\mathbf{j})$

b. John claimed that Mary or Tom left.
$\mathbf{claimed}(\mathbf{or}(\mathbf{left}\,\mathbf{m})(\mathbf{left}\,\mathbf{t}))\,\mathbf{j}$

Examples (39a) and (39b) show the behavior of coordination with and without a reset at the level of the embedded clause. In (39a), John either claims Mary left or claims Tom left; in (39b), John makes a single indeterminate claim that either Mary or Tom left. Since the interpretation in (39b) is certainly possible for this sentence, it provides additional evidence that a reset for delimiting embedded clauses must be available.

Finally, we have misplaced modifiers:

(40) a. John saw a tall sailor.
saw (**a**(**tall sailor**)) **j**

b. John saw an occasional sailor.
occasionally (**saw** (**a sailor**) **j**)

c. John claimed that an occasional sailor left.
claimed (**occasionally** (**left** (**a sailor**))) **j**

The adjective *tall* is a normal adjective and has no special control properties. The adjective *occasional*, however, takes scope over an entire clause. In (40c), in order for the leaving to be occasional rather than the claiming activity to be occasional, it is once again necessary to provide a reset delimiting the scope of the misplaced modifier within the embedded clause.

The examples above show that quantification, focus, coordination, and misplaced modifiers all require some form of delimitation at least some of the time in order to generate appropriate interpretations.

A word of warning: each of the analyses in this brief survey has grave inadequacies in the simple form presented here. In addition, many of the interactions among the phenomena behave badly in the cumulative fragment (consider, for instance, that is not possible to generate an analysis for pitch accent on just one conjunct, even thought this is something that native speakers have no trouble interpreting, as for *John only claimed that Mary or TOM left*). This is the normal state of affairs when dealing with natural language, of course; as Sapir put it, "all grammars leak". The delight comes from figuring out an equally simple grammar that performs better. In any case, I am not aware of any potential problem that could not be dealt with by a suitably-developed continuation-based analysis along the lines of the approximations offered here; in other words, I believe each of these proposals to be a viable approach.

9. CONCLUSIONS

I have suggested that continuations provide an appealing analysis of a variety of natural language phenomena. It is possible for a skeptic to claim that natural language semantics has gotten by well enough without continuations so far. To be sure, each of the phenomena discussed above have well-established treatments in the linguistic literature that do not mention continuations. In the same way, computer scientists were perfectly able to deal meaningfully with the difference between call-by-value versus call-by-name before Plotkin's CPS analysis [10]. Yet I take it that these days everyone recognizes that a full understanding of evaluation disciplines requires continuations (or else something very like continuations). It is my hope, then, that the examples in this paper, or other analyses yet to be discovered, either individually or cumulatively, will someday make continuations seem as indispensable for the description of natural language as they currently are for the theory of computation and logic.

In closing, I would like to add one more subquestion to the list of questions discussed in the introductory section.

- [**Innovation**] Are there uses for continuations in natural language that computer scientists haven't thought up yet?

Besides whatever intrinsic interest there might be in finding continuations in a natural setting, it is conceivable that once we look more closely, natural language will do interesting things with continuations that have not yet been dreamt up by theoretical computer scientists. I doubt that any of the natural language constructions treated above will seem breathtakingly new to an experienced continuation hacker; but then, I have chosen these examples precisely in order to maximize the degree to which they seem like garden-variety control operators. In the same spirit that pharmaceutical companies survey compounds harvested from tropical rain forests in hopes of finding medically useful substances unknown to laboratory scientists, we should consider that there are a lot of poorly understood languages out there—who knows what amazing control operators lurk in languages spoken in the forests of New Guinea?

10. ACKNOWLEDGMENTS

Thanks to Harry Mairson, Hayo Thielecke and Chung-chieh Shan.

11. REFERENCES

[1] C. Barker. Continuations and the nature of quantification. *Natural Language Semantics*, 10:211–242, 2002.

[2] P. de Groote. Type raising, continuations, and classical logic. In van Rooy and Stokhof [20], pages 97–101.

[3] M. Felleisen. *The Calculi of λ_v-CS Conversion: A Syntactic Theory of Control and State in Imperative Higher-Order Programming Languages.* PhD thesis, Indiana University, Aug. 1987. Also as Tech. Rep. 226, Department of Computer Science, Indiana University.

[4] M. Felleisen. The theory and practice of first-class prompts. In *POPL '88: Conference Record of the Annual ACM Symposium on Principles of Programming Languages*, pages 180–190, New York, 1988. ACM Press.

[5] T. G. Griffin. A formulae-as-types notion of control. In *POPL '90: Conference Record of the Annual ACM Symposium on Principles of Programming Languages*, pages 47–58, New York, 1990. ACM Press.

[6] H. Hendriks. *Studied Flexibility: Categories and Types in Syntax and Semantics.* PhD thesis, Institute for Logic, Language and Computation, Universiteit van Amsterdam, 1993.

[7] A. R. Meyer and M. Wand. Continuation semantics in typed lambda-calculi (summary). In R. Parikh, editor, *Logics of Programs*, number 193 in Lecture Notes in Computer Science, pages 219–224, Berlin, 1985. Springer-Verlag.

[8] R. Montague. The proper treatment of quantification in ordinary English. In R. Thomason, editor, *Formal Philosophy: Selected Papers of Richard Montague*, pages 247–270. Yale University Press, New Haven, 1974.

[9] C. R. Murthy. *Extracting Constructive Content from Classical Proofs.* PhD thesis, Department of Computer Science, Cornell University, Aug. 1990. Also as Tech. Rep. TR90-1151.

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

[11] C. Queinnec. Inverting back the inversion of control or, continuations versus page-centric programming. Rapport de Recherche LIP6 2001/007, Laboratoire d'Informatique de Paris 6, 2001.

[12] J. C. Reynolds. The discoveries of continuations. *Lisp and Symbolic Computation*, 6(3–4):233–247, 1993.

[13] C.-c. Shan. Monads for natural language semantics. In K. Striegnitz, editor, *Proceedings of the ESSLLI-2001 Student Session*, pages 285–298, Helsinki, 2001. 13th European Summer School in Logic, Language and Information.

[14] C.-c. Shan. A variable-free dynamic semantics. In van Rooy and Stokhof [20], pages 204–209.

[15] C.-c. Shan. A continuation semantics of interrogatives that accounts for Baker's ambiguity. In B. Jackson, editor, *SALT XII: Semantics and Linguistic Theory*, pages 246–265, Ithaca, 2002. Cornell University Press.

[16] C.-c. Shan. Quantifier strengths predict scopal possibilities of Mandarin Chinese *wh*-indefinites. Draft manuscript, 2003.

[17] C.-c. Shan and C. Barker. Explaining crossover and superiority as left-to-right evaluation. Draft manuscript, 2003.

[18] D. Sitaram. Handling control. In *PLDI '93: Proceedings of the ACM Conference on Programming Language Design and Implementation*, volume 28(6) of *ACM SIGPLAN Notices*, pages 147–155, New York, June 1993. ACM Press.

[19] D. Sitaram and M. Felleisen. Control delimiters and their hierarchies. *Lisp and Symbolic Computation*, 3(1):67–99, Jan. 1990.

[20] R. van Rooy and M. Stokhof, editors. *Proceedings of the 13th Amsterdam Colloquium*. Institute for Logic, Language and Computation, Universiteit van Amsterdam, 2001.

Delimited continuations in natural language

Quantification and polarity sensitivity

Chung-chieh Shan
Harvard University
33 Oxford Street
Cambridge, MA 02138 USA
ccshan@post.harvard.edu

ABSTRACT

Making a linguistic theory is like making a programming language: one typically devises a type system to delineate the acceptable utterances and a denotational semantics to explain observations on their behavior. Via this connection, the programming language concept of delimited continuations can help analyze natural language phenomena such as quantification and polarity sensitivity. Using a logical metalanguage whose syntax includes control operators and whose semantics involves evaluation order, these analyses can be expressed in direct style rather than continuation-passing style, and these phenomena can be thought of as computational side effects.

Categories and Subject Descriptors

D.3.3 [**Programming languages**]: Language constructs and features—*control structures*; J.5 [**Linguistics**]

General Terms

Languages, Theory

Keywords

Delimited continuations, control effects, natural language semantics, quantification, polarity sensitivity

1. INTRODUCTION

This paper is about computational linguistics, in the sense of applying insights from computer science to linguistics. Linguistics strives to scientifically explain empirical observations of natural language. Semantics, in particular, is concerned with phenomena such as the following. In (1) below, some sentences to the left *entail* their counterparts to the right, but others do not.

(1) Every student passed $\vdash$ Every diligent student passed
No student passed $\vdash$ No diligent student passed
A student passed $\nvdash$ A diligent student passed
Most students passed $\nvdash$ Most diligent students passed

The sentence in (2) is *ambiguous* between at least two readings. On one reading, the speaker must decline to run any spot that fails to substantiate any claims whatsoever. On another reading, there exist certain claims (anti-war ones, say) such that the speaker must decline to run any spot that fails to substantiate them.

(2) We must decline to run any spot that fails to substantiate certain claims.[1]

Finally, among the four sentences in (3), only (3a) is *acceptable.* That is, only it can be used in idealized conversation. The unacceptability of the rest is notated with asterisks.

(3) a. No student liked any course.
b. *Every student liked any course.
c. *A student liked any course.
d. *Most students liked any course.

The linguistic entailments and non-entailments in (1) are facts about English, in that only a speaker of English can make these judgments. Nevertheless, they presumably have to do with corresponding logical entailments and non-entailments: both the English speaker who judges that *Every student passed* entails *Every diligent student passed* and the Mandarin speaker who judges that *Meige xuesheng dou jige-le* entails *Meige yonggong-de xuesheng dou jige-le* rely on knowing that, if every student passed, then every diligent student passed. Thus the typical linguistic theory specifies a semantics for natural language by translating declarative sentences into logical statements with truth conditions. The linguistic entailments in (1) hold, goes the theory, because the meanings—truth conditions—of the two sentences are such that any model that verifies the former also verifies the latter. Much work in natural language semantics aims in this way, as depicted in Figure 1, to explain the horizontal by positing the vertical. This approach is reminiscent of programming language research where an ill-understood language (perhaps one with a complicating feature like ex-

Continuation Workshop 2004 Venice, Italy

[1] This sentence is part of a statement made by the cable television company Comcast after its CNN channel rejected an anti-war commercial hours before it was scheduled to air on January 28, 2003.

$$\begin{tikzcd}
\text{Every student passed} \arrow[d, squiggly] & \vdash & \text{Every diligent student passed} \arrow[d, squiggly] \\
\forall x.\, \mathsf{student}(x) \Rightarrow \mathsf{passed}(x) \arrow[d, squiggly] & \vdash & \forall x.\, \big(\mathsf{student}(x) \wedge \mathsf{diligent}(x)\big) \Rightarrow \mathsf{passed}(x) \arrow[d, squiggly] \\
\langle \text{some truth condition on models} \rangle & \vdash & \langle \text{some other truth condition on models} \rangle
\end{tikzcd}$$

Figure 1: The translation/denotation approach to natural language semantics

ceptions) is studied by translation into a simpler language (without exceptions) that is better understood.

The translation target posited in natural language semantics is often some combination of the λ-calculus and predicate logic. For example, the verb *passed* might be translated as $\lambda x.\, \mathsf{passed}(x)$. This paper argues by example that the translation target should be a logical metalanguage with first-class delimited continuations. The examples are two natural language phenomena: *quantification* by words like *every* and *most* in (1), and *polarity sensitivity* on the part of words like *any* in (3).

Quantification was first analyzed explicitly using continuations by Barker (2002). Building on that insight, this paper makes the following two contributions. First, I analyze natural language in direct style rather than in continuation-passing style. In other words, the logical metalanguage used here is one that includes control operators for delimited continuations, rather than a pure λ-calculus in which denotations need to handle continuations explicitly. Natural language is thus endowed with an operational semantics from computer science that is richer than just $\beta\eta$-reduction.

Second, I propose a new analysis of polarity sensitivity that improves upon prior theories in explaining why *No student liked any course* is acceptable but **Any student liked no course* is not. This analysis crucially relies on the notion of evaluation order from programming languages, thus elucidating the role of control effects in natural language and supporting the broader claim that linguistic phenomena can be fruitfully thought of as computational side effects.

The rest of this paper is organized as follows. In §2, I introduce a simple grammatical formalism. In §3, I describe the linguistic phenomenon of quantification and show a straw man analysis that deals with some cases but not others. I then introduce a programming language with delimited continuations and use it to improve the straw man analysis: quantification in non-subject position is treated in §4, and inverse scope is covered in §5. In §6, I turn to the linguistic phenomenon of polarity sensitivity and show how a computationally motivated notion of evaluation order improves upon previous analyses. In §7, I place these examples in a broader context and conclude.

2. A GRAMMATICAL FORMALISM

In this section, I introduce a simple grammatical formalism for use in the rest of the paper. It is a notational variant of categorial grammar (as introduced by Carpenter (1997; chapter 4), for instance).

The verb *like* usually requires an object to its right and a subject to its left.

(4) a. Alice liked CS187.
b. *Alice liked.
c. *Alice liked Bob CS187.

Intuitively, *like* is a function that takes two arguments, and the sentences (4b–c) are unacceptable due to type mismatch. We can model this formally by assigning types to the denotations of *Alice*, *CS187*, and *liked*, which we take to be atomic expressions.

(5) $[\![\text{Alice}]\!] = \mathsf{alice} : \mathsf{Thing}$

(6) $[\![\text{CS187}]\!] = \mathsf{cs187} : \mathsf{Thing}$

(7) $[\![\text{liked}]\!] = \mathsf{liked} : \mathsf{Thing} \to \mathsf{Thing} \to \mathsf{Bool}$

Here Thing is the type of individual objects, and Bool is the type of truth values or propositions. Following (justifiable) standard practice in linguistics, we let liked take its object as the first argument and its subject as the second argument. For example, in (4a), the first argument to liked is cs187, and the second argument is alice.

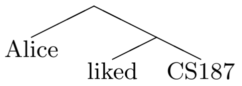
As (4a) shows, there are two ways to combine expressions. A function can take its argument either to its right (combining *liked* with *CS187*) or to its left (combining *Alice* with *liked CS187*). We denote these two cases with two infix operators: “/” for forward combination and “\” for backward combination. (The tick marks depict the direction in which a function “leans on” an argument.)

(8) $f \mathbin{/} x = f(x) : \beta \qquad \text{where } f : \alpha \to \beta,\ x : \alpha$

(9) $x \mathbin{\backslash} f = f(x) : \beta \qquad \text{where } f : \alpha \to \beta,\ x : \alpha$

We can now derive the sentence (4a)—that is, prove it to have type Bool. The derivation can be written as a tree (10) or a term (11).

(10)



(11) $[\![\text{Alice}]\!] \mathbin{\backslash} ([\![\text{liked}]\!] \mathbin{/} [\![\text{CS187}]\!]) = \mathsf{liked\ cs187\ alice} : \mathsf{Bool}$

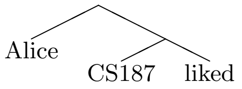
By convention, the infix operators / and \ associate to the right, so parentheses such as those in (11) above are optional.

Unfortunately, the system set up so far derives not only the acceptable sentence (4a) but also the unacceptable sentence (12), with the same meaning.

(12) *Alice CS187 liked.

The reason the system derives (12) is that the direction of function application is unconstrained: in the derivation below, *liked* takes its first (object) argument to the left, which is usually disallowed in English.

(13)



(14) $[\![\text{Alice}]\!] \mathbin{\backslash} [\![\text{CS187}]\!] \mathbin{\backslash} [\![\text{liked}]\!] = \mathsf{liked\ cs187\ alice} : \mathsf{Bool}$

To rule out this derivation of (12) in our type system, we split the function type constructor “→” into two type constructors "$\acute{\rightarrow}$" and "$\grave{\rightarrow}$", one for each direction of application. Using these new type constructors, we change the denotation of *liked* to specify that its first argument is to its right and its second argument is to its left.

(15) $$[\![\text{liked}]\!] = \mathsf{liked} : \mathsf{Thing} \mathrel{\acute{\rightarrow}} \mathsf{Thing} \mathrel{\grave{\rightarrow}} \mathsf{Bool}$$

We also revise the combination rules (8) and (9) to require different function type constructors.

(16) $$f \mathbin{/} x : \beta \qquad \text{where } f : \alpha \mathrel{\acute{\rightarrow}} \beta,\ x : \alpha$$

(17) $$x \mathbin{\backslash} f : \beta \qquad \text{where } f : \alpha \mathrel{\grave{\rightarrow}} \beta,\ x : \alpha$$

The system now rejects (12) while continuing to accept (4a), as desired.

3. QUANTIFICATION

The linguistic phenomenon of quantification is illustrated by the following sentences.

(18) a. Every student liked CS187.
b. Some student liked every course.
c. Alice consulted Bob before most meetings.

As with the previously encountered sentences, the natural language semanticist wants to translate English into logical formulas that account for entailment and other properties. More precisely, the problem is to posit translation rules that map these sentences thus. For instance, we would like to map (18a) to a formula like

(19) $$\forall x.\, \mathsf{student}(x) \Rightarrow x \mathbin{\backslash} \mathsf{liked} \mathbin{/} \mathsf{cs187} : \mathsf{Bool},$$

where the constants

(20) $$\forall : (\mathsf{Thing} \rightarrow \mathsf{Bool}) \rightarrow \mathsf{Bool}, \qquad \Rightarrow : \mathsf{Bool} \rightarrow \mathsf{Bool} \rightarrow \mathsf{Bool}$$

are drawn from the (higher-order) abstract syntax of predicate logic. To this end, what should the subject noun phrase *every student* denote? Unlike with *Alice*, there is nothing of type Thing that the quantificational noun phrase *every student* can denote and still allow the desired translation (19) to be generated. At the same time, we would like to retain the denotation that we previously computed for the verb phrase *liked CS187*, namely $\mathsf{liked} \mathbin{/} \mathsf{cs187}$. Taking these considerations into account, one way to translate (18a) to (19) is for the determiner *every* to denote

(21) $$[\![\text{every}]\!] = \acute{\lambda} r.\, \acute{\lambda} s.\, \forall x.\, r(x) \Rightarrow x \mathbin{\backslash} s$$
$$: (\mathsf{Thing} \rightarrow \mathsf{Bool}) \mathrel{\acute{\rightarrow}} (\mathsf{Thing} \mathrel{\grave{\rightarrow}} \mathsf{Bool}) \mathrel{\acute{\rightarrow}} \mathsf{Bool}.$$

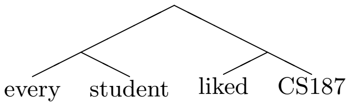
Here the *restrictor* r and the *scope* s are λ-bound variables intended to receive, respectively, the denotations of the noun *student* (of type $\mathsf{Thing} \rightarrow \mathsf{Bool}$) and the verb phrase *liked CS187* (of type $\mathsf{Thing} \mathrel{\grave{\rightarrow}} \mathsf{Bool}$). (More precisely, r and s are $\acute{\lambda}$-bound variables; the tick mark again signifies the direction of function application.) In a non-quantificational sentence like (4a), the verb phrase takes the subject as its argument; by contrast, in the quantificational sentence (18a), the subject takes the verb phrase as its argument.

Extended with the lexical entry (21) for *every*, and assuming that *student* denotes

(22) $$[\![\text{student}]\!] = \mathsf{student} : \mathsf{Thing} \rightarrow \mathsf{Bool},$$

the grammar can derive the sentence (18a).

(23)



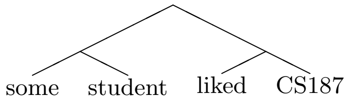
(24) $$([\![\text{every}]\!] \mathbin{/} [\![\text{student}]\!]) \mathbin{/} [\![\text{liked}]\!] \mathbin{/} [\![\text{CS187}]\!] = (19)$$

The existential determiner *some* can be analyzed similarly: let *some* denote

(25) $$[\![\text{some}]\!] = \acute{\lambda} r.\, \acute{\lambda} s.\, \exists x.\, r(x) \wedge x \mathbin{\backslash} s$$
$$: (\mathsf{Thing} \rightarrow \mathsf{Bool}) \mathrel{\acute{\rightarrow}} (\mathsf{Thing} \mathrel{\grave{\rightarrow}} \mathsf{Bool}) \mathrel{\acute{\rightarrow}} \mathsf{Bool}$$

to derive the sentence *Some student liked CS187.*

(26)



(27) $$([\![\text{some}]\!] \mathbin{/} [\![\text{student}]\!]) \mathbin{/} [\![\text{liked}]\!] \mathbin{/} [\![\text{CS187}]\!]$$
$$= \exists x.\, \mathsf{student}(x) \wedge x \mathbin{\backslash} \mathsf{liked} \mathbin{/} \mathsf{cs187} : \mathsf{Bool}$$

To summarize, we treat determiners like *every* and *some* as functions of two arguments: the restrictor and the scope of a quantifier, both functions from Thing to Bool. Such higher-order functions are a popular analysis of natural language determiners, and have been known to semanticists since Montague (1974) as *generalized quantifiers.* However, the simplistic account presented above only handles quantificational noun phrases in subject position, as in (18a) but not (18b) or (18c). For example, in (18b), neither forward nor backward combination can apply to join the verb *liked*, of type $\mathsf{Thing} \mathrel{\acute{\rightarrow}} \mathsf{Thing} \mathrel{\grave{\rightarrow}} \mathsf{Bool}$, to its object *every course*, of type $(\mathsf{Thing} \mathrel{\grave{\rightarrow}} \mathsf{Bool}) \mathrel{\acute{\rightarrow}} \mathsf{Bool}$. Yet, empirically speaking, the sentence (18b) is not only acceptable but in fact ambiguous between two available readings. This problem has prompted a great variety of supplementary proposals in the linguistics literature (Barwise and Cooper 1981; Hendriks 1993; May 1985; inter alia). The next section presents a solution using *delimited continuations.*

4. DELIMITED CONTINUATIONS

First-class continuations represent "the entire (default) future for the computation" (Kelsey, Clinger, Rees et al. 1998). Refining this concept, Felleisen (1988) introduced *delimited* continuations, which encapsulate only a prefix of that future. This paper uses *shift* and *reset* (Danvy and Filinski 1989, 1990), a popular choice of control operators for delimited continuations.

To review briefly, the shift operator (notated ξ) captures the current context of computation, removing it and making it available to the program as a function. For example, when evaluating the term

(28) $$10 \times (\xi f.\, 1 + f(2)),$$

the variable f is bound to the function that multiplies every number by 10. Thus the above expression evaluates to 21 via the following sequence of reductions. (The reduced subexpression at each step is underlined.)

(29) $$[10 \times \underline{(\xi f.\, 1 + f(2))}]$$
$$\rhd [1 + \underline{(\lambda v.\, [10 \times v])(2)}]$$
$$\rhd [1 + \underline{[10 \times 2]}] \rhd [1 + \underline{[20]}] \rhd \underline{[1 + 20]} \rhd \underline{[21]} \rhd 21$$

Term reductions are performed deterministically in applicative order: call-by-value and left-to-right.

The reset operator (notated with square brackets [ ]) delineates how far shift can reach: shift captures the current context of computation up to the closest dynamically enclosing reset. Hence “3 ×” below is out of reach.

(30) $$\begin{aligned}&\big[3 \times [10 \times (\underline{\xi f.\, 1 + f(2)})]\big]\\ &\quad\rhd \big[3 \times [1 + \underline{(\lambda v.\, [10 \times v])(2)}]\big]\\ &\quad\rhd \big[3 \times [1 + [\underline{10 \times 2}]]\big] \rhd \big[3 \times [1 + \underline{[20]}]\big] \rhd \cdots \rhd 63\end{aligned}$$

Shift and reset come with an operational semantics (illustrated by the reductions above) as well as a denotational semantics (via the CPS transform). That both kinds of semantics are available is important to linguistics, because the meanings of natural language expressions (studied in semantics) need to be related to how humans process them (studied in psycholinguistics).

Quantificational expressions in natural language can be thought of as phrases that manipulate their context. In a sentence like *Alice liked CS187* (4a), the context of *CS187* is the function mapping each thing x to the proposition that Alice liked x. Compared to the proper noun *CS187*, what is special about a quantificational expression like *every course* is that it captures its surrounding context when used.

(31) Alice liked [every course].

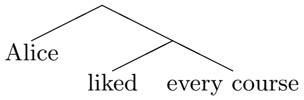
Thus, loosely speaking, the meaning of the sentence (31) no longer has the overall shape $\mathsf{alice} \mathbin{\backslash} \mathsf{liked} \mathbin{/} \cdots$ once the occurrence of *every course* is considered, much as the meaning of the program (28) no longer has the overall shape $10 \times \cdots$ once the shift expression is evaluated. Let us add shift and reset to the target language of our translation from English. We can then translate *every course* as

(32) $\llbracket\text{every course}\rrbracket = \xi s.\, \forall x.\, \mathsf{course}(x) \Rightarrow s(x) : \mathsf{Thing}^{\mathsf{Bool}}_{\mathsf{Bool}}.$

The type notation α^{δ}_{γ} here indicates an α with a control effect; the CPS transform maps it to $(\alpha \to \gamma) \to \delta$. The denotation of *every course* behaves locally as a Thing, but requires the current context to have the answer type Bool and maintains that answer type.

To see the new denotation (32) in action, let us derive the sentence (31). The type of *every course* is $\mathsf{Thing}^{\mathsf{Bool}}_{\mathsf{Bool}}$, similar to the type Thing of *CS187*, so the derivation of (31) is analogous to (10–11).

(33)



(34) $$\begin{aligned}&\big[\llbracket\text{Alice}\rrbracket \mathbin{\backslash} \llbracket\text{liked}\rrbracket \mathbin{/} \llbracket\text{every course}\rrbracket\big]\\ &\quad= [\mathsf{alice} \mathbin{\backslash} \mathsf{liked} \mathbin{/} \underline{\xi s.\, \forall x.\, \mathsf{course}(x) \Rightarrow s(x)}]\\ &\quad\rhd [\forall x.\, \mathsf{course}(x) \Rightarrow \underline{(\lambda v.\, [\mathsf{alice} \mathbin{\backslash} \mathsf{liked} \mathbin{/} v])(x)}] \rhd \cdots\\ &\quad\rhd \forall x.\, \mathsf{course}(x) \Rightarrow \mathsf{alice} \mathbin{\backslash} \mathsf{liked} \mathbin{/} x : \mathsf{Bool}\end{aligned}$$

Like the straw man analysis in §3, the denotation in (32) generalizes to determiners other than *every*: we can abstract the noun *course* out of *every course*, and deal with *some student* similarly.

(35) $$\llbracket\text{every}\rrbracket = \lambda' r.\, \xi s.\, \forall x.\, r(x) \Rightarrow s(x),$$

(36) $$\begin{aligned}\llbracket\text{some}\rrbracket &= \lambda' r.\, \xi s.\, \exists x.\, r(x) \wedge s(x)\\ &\;:\; (\mathsf{Thing} \to \mathsf{Bool}) \acute{\to} \mathsf{Thing}^{\mathsf{Bool}}_{\mathsf{Bool}}\end{aligned}$$

(We require here that the restrictor r have the type $\mathsf{Thing} \to \mathsf{Bool}$, not a type of the form $\mathsf{Thing} \to \mathsf{Bool}^{\delta}_{\gamma}$, so r cannot incur control effects when applied to x. Any control effect in the restrictor, such as induced by the quantificational noun phrase *a company* in the sentence *Every representative of a company left*, must be contained within reset.)

More importantly, unlike the straw man analysis, the new analysis works uniformly for quantificational expressions in subject, object, and other positions, such as in (18a–c). Intuitively, this is because shift captures the context of an expression no matter how deeply it is embedded in the sentence.[2] By adding control operators for delimited continuations to our logical metalanguage, we arrive at an analysis of quantification with greater empirical coverage.

Figure 2 shows a logical metalanguage that formalizes the basic ideas presented above. It is in this language that denotations on this page are written and reduced. Refining Danvy and Filinski's original shift-reset language, we distinguish between *pure* and *impure* expressions. An impure expression may incur control effects when evaluated, whereas a pure expression only incurs control effects contained within reset (Danvy and Hatcliff 1992, 1994; Nielsen 2001; Thielecke 2003). This distinction is reflected in the typing judgments: an impure judgment

(37) $$\Gamma \vdash E : \alpha^{\delta}_{\gamma}$$

not only gives a type α for E itself but also specifies two answer types γ and δ. By contrast, a pure judgment

(38) $$\Gamma \vdash E : \alpha$$

only gives a type α for E itself. As can be seen in the Lift rule, pure expressions are polymorphic in the answer type.

As mentioned in §2, the use of directionality in function types to control word order is not new in linguistics, but the use of delimited control operators to analyze quantification is. It turns out that we can tie the potential presence of control effects in function bodies to directionality. That is, only directional functions—those whose types are decorated with tick marks—are potentially impure; all non-directional functions we need to deal with, including contexts captured by shift, are pure. Another link between directionality and control effects is that the $\acute{\to}$E and $\grave{\to}$E rules for directional function application are not merely mirror images of each other: the answer types γ_0 through γ_3 are chained differently through the premises. This is due to left-to-right evaluation.

Having made the distinction between pure and impure expressions, we require in our Shift rule that the body of a shift expression be pure. This change from Danvy and Filinski's original system simplifies the type system and the CPS transform, but a shift expression $\xi f.\, E$ in their language may need to be rewritten here to $\xi f.\, [E]$.

The CPS transform for the metalanguage follows from the typing rules and is standard; it supplies a denotational semantics. The operational semantics for the metalanguage specifies a computation relation between complete terms; it is also standard and shown in Figure 3.

The present analysis is almost, but not quite, the direct-style analogue of Barker's (2002) CPS analysis. Put in direct-style terms, Barker's function bodies are always pure, whereas function bodies here can harbor control effects. In

[2] No matter how deep, that is, up to the closest dynamically enclosing reset. Control delimiters correspond to *islands* in natural language (Barker 2002).

Directions $\Delta ::= / \mid \backslash$

Types $\alpha, \beta, \gamma, \delta ::= \mathsf{Thing} \mid \mathsf{Bool} \mid \alpha \rightarrow \beta \mid \alpha \xrightarrow{\Delta} \beta^{\delta}_{\gamma}$

Antecedents $\Gamma ::= x_1 : \alpha_1, \ldots, x_n : \alpha_n$

Terms $E, F ::= c \mid x \mid \lambda x.\, E \mid \lambda^{\Delta} x.\, E \mid FE \mid F / E \mid E \backslash F \mid [E] \mid \xi f.\, E$

Constants $c : \alpha$

$$\frac{}{\forall : (\mathsf{Thing} \rightarrow \mathsf{Bool}) \rightarrow \mathsf{Bool}} \qquad \frac{}{\exists : (\mathsf{Thing} \rightarrow \mathsf{Bool}) \rightarrow \mathsf{Bool}}$$

$$\frac{}{\Rightarrow : \mathsf{Bool} \rightarrow \mathsf{Bool} \rightarrow \mathsf{Bool}} \qquad \frac{}{\wedge : \mathsf{Bool} \rightarrow \mathsf{Bool} \rightarrow \mathsf{Bool}}$$

$$\frac{}{\mathsf{student} : \mathsf{Thing} \rightarrow \mathsf{Bool}} \qquad \frac{}{\mathsf{liked} : \mathsf{Thing} \overset{\prime}{\rightarrow} (\mathsf{Thing} \overset{\backprime}{\rightarrow} \mathsf{Bool}^{\delta}_{\delta})^{\gamma}_{\gamma}} \qquad \cdots$$

Pure expressions $\Gamma \vdash E : \alpha$

$$\frac{c : \alpha}{\Gamma \vdash c : \alpha}\,\text{Const} \qquad \frac{}{\Gamma, x : \alpha \vdash x : \alpha}\,\text{Var} \qquad \frac{\Gamma \vdash E : \alpha^{\beta}_{\alpha}}{\Gamma \vdash [E] : \beta}\,\text{Reset}$$

$$\frac{\Gamma, x : \alpha \vdash E : \beta}{\Gamma \vdash \lambda x.\, E : \alpha \rightarrow \beta}\,{\rightarrow}\text{I} \qquad \frac{\Gamma, x : \alpha \vdash E : \beta^{\delta}_{\gamma}}{\Gamma \vdash \lambda^{\Delta} x.\, E : \alpha \xrightarrow{\Delta} \beta^{\delta}_{\gamma}}\,\xrightarrow{\Delta}\text{I} \qquad \frac{\Gamma \vdash F : \alpha \rightarrow \beta \quad \Gamma \vdash E : \alpha}{\Gamma \vdash FE : \beta}\,{\rightarrow}\text{E}$$

Impure expressions $\Gamma \vdash E : \alpha^{\delta}_{\gamma}$

$$\frac{\Gamma \vdash E : \alpha}{\Gamma \vdash E : \alpha^{\gamma}_{\gamma}}\,\text{Lift} \qquad \frac{\Gamma, f : \alpha \rightarrow \gamma \vdash E : \delta}{\Gamma \vdash \xi f.\, E : \alpha^{\delta}_{\gamma}}\,\text{Shift}$$

$$\frac{\Gamma \vdash F : (\alpha \overset{\prime}{\rightarrow} \beta^{\gamma 2}_{\gamma 3})^{\gamma 0}_{\gamma 1} \quad \Gamma \vdash E : \alpha^{\gamma 1}_{\gamma 2}}{\Gamma \vdash F / E : \beta^{\gamma 0}_{\gamma 3}}\,\overset{\prime}{\rightarrow}\text{E} \qquad \frac{\Gamma \vdash E : \alpha^{\gamma 0}_{\gamma 1} \quad \Gamma \vdash F : (\alpha \overset{\backprime}{\rightarrow} \beta^{\gamma 2}_{\gamma 3})^{\gamma 1}_{\gamma 2}}{\Gamma \vdash E \backslash F : \beta^{\gamma 0}_{\gamma 3}}\,\overset{\backprime}{\rightarrow}\text{E}$$

Figure 2: A logical metalanguage with directionality and delimited control operators

Values $V ::= U \mid \lambda x.\, E \mid \lambda^{\Delta} x.\, E$

Unknowns $U ::= c \mid UV \mid U / V \mid V \backslash U$

Contexts $C\langle\ \rangle ::= \langle\ \rangle \mid (C\langle\ \rangle)E \mid C\langle\ \rangle / E \mid C\langle\ \rangle \backslash F$
$\mid V(C\langle\ \rangle) \mid V / C\langle\ \rangle \mid V \backslash C\langle\ \rangle$

Metacontexts $D\langle\ \rangle ::= \langle\ \rangle \mid C\langle [D\langle\ \rangle] \rangle$

Computations $E \rhd E'$

$$\begin{aligned}
D\langle C\langle \underline{(\lambda x.\, E)V} \rangle\rangle &\rhd D\langle C\langle E\{x \mapsto V\}\rangle\rangle \\
D\langle C\langle \underline{(\lambda^{\prime} x.\, E) / V} \rangle\rangle &\rhd D\langle C\langle E\{x \mapsto V\}\rangle\rangle \\
D\langle C\langle \underline{V \backslash (\lambda^{\backprime} x.\, E)} \rangle\rangle &\rhd D\langle C\langle E\{x \mapsto V\}\rangle\rangle \\
D\langle C\langle \underline{[V]} \rangle\rangle &\rhd D\langle C\langle V \rangle\rangle \\
D\langle C\langle \underline{\xi f.\, E} \rangle\rangle &\rhd D\langle E\{f \mapsto \lambda x.\, [C\langle x \rangle]\}\rangle
\end{aligned}$$

Figure 3: Reductions for the logical metalanguage

other words, function bodies are allowed to shift, as in the determiner denotations in (35) and (36). By contrast, Barker uses *choice functions* to assign meanings to determiners.

5. QUANTIFIER SCOPE AMBIGUITY

Of course, natural language phenomena are never as simple as a couple of programming language control operators. Quantification is no exception, so to speak. For example, the sentence *Some student liked every course* (18b) is ambiguous between the following two readings.

(39) $\exists x.\, \mathsf{student}(x) \wedge \forall y.\, \mathsf{course}(y) \Rightarrow x \backslash \mathsf{liked} / y$

(40) $\forall y.\, \mathsf{course}(y) \Rightarrow \exists x.\, \mathsf{student}(x) \wedge x \backslash \mathsf{liked} / y$

In the *surface scope* reading (39), *some* takes scope over *every.* In the *inverse scope* reading (40), *every* takes scope over *some.* Given that evaluation takes place from left to right, the shift for *some student* is evaluated before the shift for *every course.* Our grammar thus predicts the surface scope reading but not the inverse scope reading. This prediction can be seen in the first few reductions of the (unique) derivation for (18b):

(41) $$\begin{aligned}
&\big[(\llbracket\text{some}\rrbracket / \llbracket\text{student}\rrbracket) \backslash \llbracket\text{liked}\rrbracket / \llbracket\text{every}\rrbracket / \llbracket\text{course}\rrbracket\big] \\
&= \Big[\Big(\underline{(\lambda^{\prime} r.\, \xi s.\, \exists x.\, r(x) \wedge s(x)) / \mathsf{student}}\Big) \\
&\qquad \backslash\, \mathsf{liked} / \big((\lambda^{\prime} r.\, \xi s.\, \forall y.\, r(y) \Rightarrow s(y)) / \mathsf{course}\big)\Big] \\
&\rhd \Big[\Big(\underline{\xi s.\, \exists x.\, \mathsf{student}(x) \wedge s(x)}\Big) \\
&\qquad \backslash\, \mathsf{liked} / \big((\lambda^{\prime} r.\, \xi s.\, \forall y.\, r(y) \Rightarrow s(y)) / \mathsf{course}\big)\Big] \\
&\rhd \Big[\exists x.\, \mathsf{student}(x) \wedge \Big(\lambda v.\, \big[v \\
&\qquad \backslash\, \mathsf{liked} / \big((\lambda^{\prime} r.\, \xi s.\, \forall y.\, r(y) \Rightarrow s(y)) / \mathsf{course}\big)\big]\Big)(x)\Big]
\end{aligned}$$

Regardless of what evaluation order we specify, as long as our rules for semantic translation remain deterministic, they will only generate one reading for the sentence. Hence our theory fails to predict the ambiguity of the sentence (18b).

To better account for the data, we need to introduce some sort of nondeterminism into our theory. There are two natural ways to proceed. First, we can allow arbitrary evaluation order, not just left-to-right. This change would render our term calculus nonconfluent, a result unwelcome for most programming language researchers but welcome for us in light of the ambiguous natural language sentence (18b). This route has been pursued with some success by Barker (2002) and de Groote (2001). However, there are empirical reasons to maintain left-to-right evaluation, one of which appears in §6.

A second way to introduce nondeterminism is to maintain left-to-right evaluation but generalize shift and reset to a *hierarchy* of control operators (Barker 2000; Danvy and Filinski 1990; Shan and Barker 2003), leaving it unspecified at which level on the hierarchy each quantificational phrase shifts. Following Danvy and Filinski, we extend our logical metalanguage by superscripting every shift expression and pair of reset brackets with a nonnegative integer to indicate a level on the control hierarchy. Level 0 is the highest level (not the lowest). When a shift expression at level n is evaluated, it captures the current context of computation up to the closest dynamically enclosing reset at level n or higher (smaller). For example, whereas the expression

(42) $$\big[3 \times [10 \times (\xi^5 f.\, 1 + f(2))]^5\big]^0$$

evaluates to 63 as in (30), the expression

(43) $$\big[3 \times [10 \times (\xi^3 f.\, 1 + f(2))]^5\big]^0$$

evaluates to $1 + 3 \times 10 \times 2$, or 61. The superscripts can be thought of "strength levels" for shifts and resets.

Danvy and Filinski (1990) give a denotational semantics for multiple levels of delimited control using continuations of higher-order type. We can take advantage of that work in our quantificational denotations (35–36) by letting them shift at any level. The ambiguity of (18b) is then predicted as follows. Suppose that *some student* shifts at level m and *every course* shifts at level n.

(44) Some^m student liked every^n course.

If $m \leq n$, the surface scope reading (39) results. If $m > n$, the inverse scope reading (40) results. In general, a quantifier that shifts at a higher level always scopes over another that shifts on a lower level, regardless of which one is evaluated first. This way, evaluation order does not determine scoping possibilities among quantifiers in a sentence unless two quantifiers happen to shift at the same level.

To summarize the discussion so far, whether we introduce nondeterministic evaluation order or a hierarchy of delimited control operators, we can account for the ambiguity of the sentence (18b), as well as more complicated cases of quantification in English and Mandarin (Shan 2003). For example, both the nondeterministic evaluation order approach and the control hierarchy approach predict correctly that the sentence below, with three quantifiers, is 5-way ambiguous.

(45) Every representative of a company saw most samples.

Despite the fact that there are three quantifiers in this sentence and $3! = 6$, this sentence has only 5 readings. Because *a company* occurs within the restrictor of *every representative of a company*, it is incoherent for *every* to scope over *most* and *most* over *a*. The reason neither approach generates such a reading can be seen in the denotation of *every* in (35): "$\forall x$" is located immediately above "$\Rightarrow$" in the abstract syntax, with no intervening control delimiter, so no control operator can insert any material (such as *most*-quantification over samples) in between.

There exist in the computational linguistics literature algorithms for computing the possible quantifier scopings of a sentence like (45) (Hobbs and Shieber 1987; followed by Lewin 1990; Moran 1988). Having related quantifier scoping to control operators, we gain a denotational understanding of these algorithms that accords with our theoretical intuitions and empirical observations.

An extended logical metalanguage with an infinite hierarchy of control operators is shown in Figure 4. This system is more complex than the one in Figure 2 in two ways. First, instead of making a binary distinction between pure and impure expressions, we use a number to measure "how pure" each expression is. An expression is pure up to level n if it only incurs control effects at levels above n when evaluated. Pure expressions are the special case when $n = 0$. The purity level of an expression is reflected in its typing judgment: a judgment

(46) $$\Gamma \vdash E : \alpha!n$$

states that the expression E is pure up to level n. Here $\alpha!n$ is a *computation type* with n levels: as defined in the figure, it consists of $2^{n+1} - 1$ value types that together specify how a computation that is pure up to level n affects answer types between levels 0 and $n - 1$. In the special case where $n = 1$, the computation type $\alpha!1$ is of the familiar form α_γ^δ.

In the previous system in Figure 2, directional functions are always impure (that is, pure up to level 1) while non-directional functions are always pure (that is, pure up to level 0). In the current system, both kinds of functions declare in their types up to what level their bodies are pure. For example, the determiners *every* and *some*, now allowed to shift at any level, both have not just the type

(47) $$(\mathsf{Thing} \to \mathsf{Bool}) \acute{\to} \mathsf{Thing}_{\mathsf{Bool}_{\gamma!n}^{\delta!n}}^{\mathsf{Bool}_{\gamma!n}^{\delta!n}},$$

but also the type

(48) $$(\mathsf{Thing} \to \mathsf{Bool}_{\gamma 1!n}^{\gamma 0!n}) \acute{\to} \mathsf{Thing}_{\mathsf{Bool}_{\gamma 2!n}^{\gamma 1!n}}^{\mathsf{Bool}_{\gamma 2!n}^{\gamma 0!n}}$$

(but see the second technical complication below). As the argument type $\mathsf{Thing} \to \mathsf{Bool}_{\gamma 1!n}^{\gamma 0!n}$ above shows, the first argument to these determiners, the restrictor, is non-directional yet can be impure (that is, pure up to level $n + 1$).

To traverse the control hierarchy, we add a new Reset rule, which makes an expression more pure, and a new Lift rule, which makes an expression less pure. (Consecutive nested resets like $\big[[E]^{n+1}\big]^n$ can be abbreviated to $[E]^n$ without loss of coherence.)

A second complication in this system, in contrast to Figure 2, is that we can no longer encode logical quantification using a higher-order constant like $\forall : (\mathsf{Thing} \to \mathsf{Bool}) \to \mathsf{Bool}$, because such a constant requires its argument—the logical formula to be quantified—to be a pure function. This requirement is problematic because it is exactly the impurity of quantified logical formulas that underlies this account of quantifier scope ambiguity. On one hand, we want to quantify logical formulas that are impure; on the other hand, we want to rule out expressions like

(49) $$\forall x.\, \xi f.\, x,$$

where the logical variable x "leaks" illicitly into the surrounding context. This issue is precisely the problem of classifying open and closed terms in staged programming (see Taha and Nielsen 2003 and references therein): the types Thing and Bool really represent not individuals or truth values but staged programs that compute individuals and truth values. For this paper, we adopt the simplistic solution of adjoining to these types a set of free logical variables for

Directions $\Delta ::= / \mid \backslash$

Value types $\alpha, \beta, \gamma, \delta ::= \mathsf{Thing}\{\vec{p}\} \mid \mathsf{Bool}\{\vec{p}\} \mid \alpha \to \beta!n \mid \alpha \xrightarrow{\Delta} \beta!n$

Computation types $\alpha!0 ::= \alpha, \qquad \alpha!(n+1) ::= \alpha^{\delta!n}_{\gamma!n}$

Antecedents $\Gamma ::= x_1 : \alpha_1, \dots, x_n : \alpha_n$

Terms $E, F ::= c \mid x \mid \lambda^n x.\, E \mid \lambda^{\Delta n} x.\, E \mid FE \mid F \mathbin{/} E \mid E \mathbin{\backslash} F \mid [E]^n \mid \xi^n f.\, E$

Constants $c : \alpha$

$$\frac{}{p : \mathsf{Thing}\{p\}} \qquad \frac{}{\forall p : \mathsf{Bool}\{p, \vec{q}\} \to \mathsf{Bool}\{\vec{q}\}} \qquad \frac{}{\exists p : \mathsf{Bool}\{p, \vec{q}\} \to \mathsf{Bool}\{\vec{q}\}}$$

$$\frac{}{\Rightarrow : \mathsf{Bool}\{\vec{p}\} \to \mathsf{Bool}\{\vec{q}\} \to \mathsf{Bool}\{\vec{p} \cup \vec{q}\}} \qquad \frac{}{\wedge : \mathsf{Bool}\{\vec{p}\} \to \mathsf{Bool}\{\vec{q}\} \to \mathsf{Bool}\{\vec{p} \cup \vec{q}\}}$$

$$\frac{}{\mathsf{student} : \mathsf{Thing}\{\vec{p}\} \to \mathsf{Bool}\{\vec{p}\}} \qquad \frac{}{\mathsf{liked} : \mathsf{Thing}\{\vec{p}\} \acute{\rightarrow} (\mathsf{Thing}\{\vec{q}\} \grave{\rightarrow} \mathsf{Bool}\{\vec{p} \cup \vec{q}\}^{\delta!m}_{\delta!m})^{\gamma!n}_{\gamma!n}} \qquad \cdots$$

Expressions $\Gamma \vdash E : \alpha!n$

$$\frac{c : \alpha}{\Gamma \vdash c : \alpha}\,\text{Const} \qquad \frac{}{\Gamma, x : \alpha \vdash x : \alpha}\,\text{Var} \qquad \frac{\Gamma \vdash E : \alpha^{\beta}_{\alpha}}{\Gamma \vdash [E]^0 : \beta}\,\text{Reset} \qquad \frac{\Gamma \vdash E : \alpha^{\beta!(n+1)}_{\alpha^{\gamma!n}_{\gamma!n}}}{\Gamma \vdash [E]^{n+1} : \beta!(n+1)}\,\text{Reset}$$

$$\frac{\Gamma, x : \alpha \vdash E : \beta!n}{\Gamma \vdash \lambda^{(\Delta)} x.\, E : \alpha \xrightarrow{(\Delta)} \beta!n}\,\xrightarrow{(\Delta)}\text{I} \qquad \frac{\Gamma \vdash F : \alpha \to \beta!n \quad \Gamma \vdash E : \alpha}{\Gamma \vdash FE : \beta!n}\,\to\text{E}$$

$$\frac{\Gamma \vdash E : \alpha}{\Gamma \vdash E : \alpha^{\gamma}_{\gamma}}\,\text{Lift} \qquad \frac{\Gamma \vdash E : \alpha^{\gamma 0!n}_{\gamma 1!n}}{\Gamma \vdash E : \alpha^{\beta^{\gamma 0!n}_{\gamma 2!n}}_{\beta^{\gamma 1!n}_{\gamma 2!n}}}\,\text{Lift} \qquad \frac{\Gamma, f : \alpha \to \gamma!n \vdash E : \delta!n}{\Gamma \vdash \xi^n f.\, E : \alpha^{\delta!n}_{\gamma!n}}\,\text{Shift}$$

$$\frac{\Gamma \vdash F : (\alpha \acute{\rightarrow} \beta^{\gamma 2!n}_{\gamma 3!n})^{\gamma 0!n}_{\gamma 1!n} \quad \Gamma \vdash E : \alpha^{\gamma 1!n}_{\gamma 2!n}}{\Gamma \vdash F \mathbin{/} E : \beta^{\gamma 0!n}_{\gamma 3!n}}\,\acute{\rightarrow}\text{E} \qquad \frac{\Gamma \vdash E : \alpha^{\gamma 0!n}_{\gamma 1!n} \quad \Gamma \vdash F : (\alpha \grave{\rightarrow} \beta^{\gamma 2!n}_{\gamma 3!n})^{\gamma 1!n}_{\gamma 2!n}}{\Gamma \vdash E \mathbin{\backslash} F : \beta^{\gamma 0!n}_{\gamma 3!n}}\,\grave{\rightarrow}\text{E}$$

$$\frac{\Gamma \vdash F : (\alpha \acute{\rightarrow} \beta) \quad \Gamma \vdash E : \alpha}{\Gamma \vdash F \mathbin{/} E : \beta}\,\acute{\rightarrow}\text{E} \qquad \frac{\Gamma \vdash E : \alpha \quad \Gamma \vdash F : (\alpha \grave{\rightarrow} \beta)}{\Gamma \vdash E \mathbin{\backslash} F : \beta}\,\grave{\rightarrow}\text{E}$$

Figure 4: Extending the logical metalanguage to a hierarchy of control operators

tracking purposes, denoted $p, q, \dots$. Unfortunately, we also need to stipulate that these logical variables be freshly α-renamed ("created by gensym") for each occurrence of a quantifier in a sentence.

As before, the CPS transform and reductions for this metalanguage are standard; the latter appears in Figure 5.

The present analysis is almost, but not quite, the direct-style analogue of Shan and Barker's (2003) CPS analysis, even though both use a control hierarchy. Each level in Shan and Barker's hierarchy is intuitively a staged computation produced at one level higher. More concretely, a computation type with n levels in that system has the shape

(50) $(\alpha^{\gamma 0}_{\gamma 1})^{\delta 0}_{\delta 1}$

rather than the shape

(51) $\alpha^{\gamma 0^{\delta 0}_{\delta 1}}_{\gamma 1^{\delta 2}_{\delta 3}}.$

The issue above of how to encode logical quantification over impure formulas receives a more satisfactory treatment in Shan and Barker's system: no stipulation of α-renaming is necessary, because there is no analogue of (49) to prohibit. The relation between that system and staged programming with effects has yet to be explored.

6. POLARITY SENSITIVITY

Because the analysis so far focuses on the truth-conditional meaning of quantifiers, it equates the determiners *a* and *some*—both are existential quantifiers with the type and denotation in (36). Furthermore, sentences like *Has anyone arrived?* suggest that the determiner *any* also means the same thing as *a* and *some*. To the contrary, though, the determiners *a*, *some*, and *any* are not always interchangeable in their existential usage. The sentences and readings in (52) show that they take scope differently relative to negation (in these cases the quantifier *no*).

(52) a. No student liked some course. (unambiguous $\exists\neg$)
b. No student liked a course. (ambiguous $\neg\exists$, $\exists\neg$)
c. No student liked any course. (unambiguous $\neg\exists$)
d. Some student liked no course. (unambiguous $\exists\neg$)
e. A student liked no course. (ambiguous $\neg\exists$, $\exists\neg$)
f. *Any student liked no course. (unacceptable)

The determiner *any* is a *negative polarity item*: to a first approximation, it can occur only in *downward-entailing* contexts, such as under the scope of a *monotonically decreasing* quantifier (Ladusaw 1979). A quantifier q, of type $(\mathsf{Thing} \to \mathsf{Bool}) \to \mathsf{Bool}$, is monotonically decreasing just in case

(53) $\forall s_1.\, \forall s_2.\, \big(\forall x.\, s_2(x) \Rightarrow s_1(x)\big) \Rightarrow q(s_1) \Rightarrow q(s_2).$

The quantificational noun phrases *no student* and *no course* are monotonically decreasing since, for instance, if no student liked any course in general, then no student liked any computer science course in particular.

Whereas *any* is a negative polarity item, *some* is a *positive polarity item.* Roughly speaking, *some* is allergic to down-

Values $\qquad V ::= U \mid \lambda x.\,E \mid \lambda^{\Delta} x.\,E$

Unknowns $\qquad U ::= c \mid UV \mid U \mathbin{/} V \mid V \mathbin{\backslash} U$

Contexts $\qquad C\langle\ \rangle ::= \langle\ \rangle \mid (C\langle\ \rangle)E \mid C\langle\ \rangle \mathbin{/} E \mid C\langle\ \rangle \mathbin{\backslash} F \mid V(C\langle\ \rangle) \mid V \mathbin{/} C\langle\ \rangle \mid V \mathbin{\backslash} C\langle\ \rangle$

Metacontexts at level n $\qquad D^n\langle\ \rangle ::= \langle\ \rangle \mid D^{n+1}\langle D^{n+2}\langle \cdots \langle C\langle [D^n\langle\ \rangle]^n\rangle\rangle \cdots\rangle\rangle$

Computations $E \rhd E'$

$$\begin{aligned}
D^0\langle D^1\langle \cdots \langle C\langle \underline{(\lambda x.\,E)V}\rangle\rangle \cdots\rangle\rangle &\quad\rhd\quad D^0\langle D^1\langle \cdots \langle C\langle E\{x \mapsto V\}\rangle\rangle \cdots\rangle\rangle \\
D^0\langle D^1\langle \cdots \langle C\langle \underline{(\acute{\lambda} x.\,E) \mathbin{/} V}\rangle\rangle \cdots\rangle\rangle &\quad\rhd\quad D^0\langle D^1\langle \cdots \langle C\langle E\{x \mapsto V\}\rangle\rangle \cdots\rangle\rangle \\
D^0\langle D^1\langle \cdots \langle C\langle \underline{V \mathbin{\backslash} (\grave{\lambda} x.\,E)}\rangle\rangle \cdots\rangle\rangle &\quad\rhd\quad D^0\langle D^1\langle \cdots \langle C\langle E\{x \mapsto V\}\rangle\rangle \cdots\rangle\rangle \\
D^0\langle D^1\langle \cdots \langle C\langle \underline{[V]}\rangle\rangle \cdots\rangle\rangle &\quad\rhd\quad D^0\langle D^1\langle \cdots \langle C\langle V\rangle\rangle \cdots\rangle\rangle \\
D^0\langle D^1\langle \cdots \langle D^n\langle D^{n+1}\langle \cdots \langle C\langle \underline{\xi^n f.\,E}\rangle\rangle \cdots\rangle\rangle\rangle \cdots\rangle\rangle &\quad\rhd\quad D^0\langle D^1\langle \cdots \langle D^n\langle E\{f \mapsto \lambda x.\,[D^{n+1}\langle \cdots \langle C\langle x\rangle\rangle \cdots\rangle]^n\}\rangle\rangle \cdots\rangle\rangle
\end{aligned}$$

Figure 5: Reductions for the extended logical metalanguage

ward-entailing contexts (especially those with an overtly negative word like *no*). These generalizations regarding polarity items cover the data in (52a–e): in principle, goes the theory, all these sentences are ambiguous between two scopings, but the polarity sensitivity of *some* and *any* rule out one scoping each in (52a), (52c), (52d), and (52f). These four sentences are thus predicted to be unambiguous, but it remains unclear why (52f) is downright unacceptable.

In the type-theoretic tradition of linguistics, polarity sensitivity is typically implemented by splitting the answer type Bool into several types, each a different functor applied to Bool, that are related by subtyping (Bernardi 2002; Bernardi and Moot 2001; Fry 1999). For instance, to differentiate the determiners in (52) from each other in our formalism, we can add the types BoolPos and BoolNeg alongside Bool, such that both are supertypes of Bool (but not of each other).

(54) $$\frac{}{\mathsf{Bool} \leq \mathsf{BoolPos}} \qquad \frac{}{\mathsf{Bool} \leq \mathsf{BoolNeg}}$$

We also extend the subtyping relation between (value and computation) types with the usual closure rules, and allow implicit coercion from a subtype to a supertype.

(55) $$\frac{}{\alpha \leq \alpha} \qquad \frac{\alpha' \leq \alpha \quad \beta!n \leq \beta'!n}{\alpha \xrightarrow{(\Delta)} \beta!n \leq \alpha' \xrightarrow{(\Delta)} \beta'!n}$$

(56) $$\frac{\alpha' \leq \alpha \quad \gamma'!n \leq \gamma!n \quad \delta!n \leq \delta'!n}{\alpha_{\gamma!n}^{\delta!n} \leq \alpha'^{\delta'!n}_{\gamma'!n}}$$

(57) $$\frac{\Gamma \vdash E : \alpha!n \quad \alpha!n \leq \beta!n}{\Gamma \vdash E : \beta!n}\,\text{Sub}$$

We then add a side condition to Reset, requiring that the produced answer type be Bool or BoolPos, not BoolNeg.

(58) $$\frac{\Gamma \vdash E : \alpha_{\alpha}^{\beta}}{\Gamma \vdash [E] : \beta}\,\text{Reset} \qquad \text{where } \beta \leq \mathsf{BoolPos}$$

(59) $$\frac{\Gamma \vdash E : \alpha^{\beta!(n+1)}_{\alpha^{\gamma!n}_{\gamma!n}}}{\Gamma \vdash [E] : \beta!(n+1)}\,\text{Reset} \qquad \text{where } \beta \leq \mathsf{BoolPos}$$

Finally, we refine the types of our determiners from (47) to

(60) $$[\![\text{no}]\!] : (\mathsf{Thing} \to \mathsf{Bool}) \mathrel{\acute{\to}} \mathsf{Thing}^{\mathsf{Bool}^{\delta!n}_{\gamma!n}}_{\mathsf{BoolNeg}^{\delta!n}_{\gamma!n}},$$

(61) $$[\![\text{some}]\!] : (\mathsf{Thing} \to \mathsf{Bool}) \mathrel{\acute{\to}} \mathsf{Thing}^{\mathsf{BoolPos}^{\delta!n}_{\gamma!n}}_{\mathsf{BoolPos}^{\delta!n}_{\gamma!n}},$$

(62) $$[\![\text{a}]\!] : (\mathsf{Thing} \to \mathsf{Bool}) \mathrel{\acute{\to}} \mathsf{Thing}^{\mathsf{Bool}^{\delta!n}_{\gamma!n}}_{\mathsf{Bool}^{\delta!n}_{\gamma!n}},$$

(63) $$[\![\text{any}]\!] : (\mathsf{Thing} \to \mathsf{Bool}) \mathrel{\acute{\to}} \mathsf{Thing}^{\mathsf{BoolNeg}^{\delta!n}_{\gamma!n}}_{\mathsf{BoolNeg}^{\delta!n}_{\gamma!n}}.$$

The chain of answer-type transitions from one quantificational expression to the next acts as a finite-state automaton, shown in Figure 6. The states of the automaton are the three supertypes of Bool; the ϵ-transitions are the two subtyping relations in (54); and the non-ϵ transitions are the determiners in (60–63).

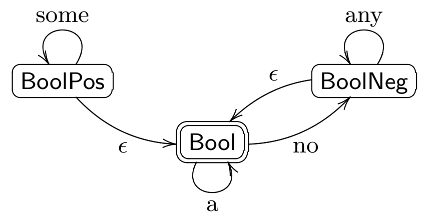


Figure 6: An automaton of answer-type transitions

This three-state machine enforces polarity constraints as follows. Any valid derivation for a sentence assigns each of its quantifiers to shift at a certain level in the control hierarchy. For each level, the quantifiers at that level—in the order in which they are evaluated—must form

- either a path from BoolPos to Bool in the state machine, in other words a string of determiners matching the regular expression "some* (a | no any*)*";
- or a path from Bool to Bool in the state machine, in other words a string of determiners matching the regular expression "(a | no any*)*".

Furthermore, the BoolPos-to-Bool levels in the hierarchy must all be higher than the Bool-to-Bool levels. Every assignment of quantifiers to levels that satisfies these conditions gives a reading for the sentence, in which quantifiers at higher levels scope wider, and, among quantifiers at the same level, ones evaluated earlier scope wider.

Consider now the two alternative ways to characterize scope ambiguity suggested in §5. The first approach is to allow arbitrary evaluation order (and use a degenerate control hierarchy of one level only). If we take this route, we can account for all of the acceptability and ambiguity judgments in (52a–e), but we cannot distinguish the acceptable sentence (52c) from the unacceptable (52f). In other words, it would be a mystery how the acceptability of a sentence hinges on the linear order in which the quantifiers *no* and *any* appear. This mystery has been noted by Ladusaw (1979; §9.2) and Fry (1999; §8.2) as a defect in current accounts of polarity sensitivity.

The second approach, using a control hierarchy with multiple levels, fares better by comparison.[3] We can stick to left-to-right evaluation, under which—as desired—an *any* must be preceded by a *no* that scopes over it with no intervening *a* or *some*. Indeed, the variations in ambiguity and acceptability among sentences in (52) are completely captured. For intuition, we can imagine that the hearer of a sentence must first process the trigger for a downward-entailing context, like *no*, before it makes sense to process a negative polarity item, like *any*.[4] Intuition aside, the programming-language notion of evaluation order provides the syntactic hacker of formal types with a new tool with which to capture observed regularities in natural language.

7. LINGUISTIC SIDE EFFECTS

This paper outlines how quantification and polarity sensitivity in natural language can be modeled using delimited continuations. These two examples support my claim that the formal theory and computational intuition we have for continuations can help us construct, understand, and maintain linguistic theories. To be sure, this work is far from the first time insights from programming languages are applied to natural language:

- It has long been noted that the intensional logic in which Montague grammar is couched can be understood computationally (Hobbs and Rosenschein 1978; Hung and Zucker 1991).
- *Dynamic semantics* (Groenendijk and Stokhof 1991), which relates anaphora and discourse in natural languages to nondeterminism and mutable state in programming languages (van Eijck 1998), has been applied to a variety of natural language phenomena, such as verb-phrase ellipsis (van Eijck and Francez 1995; Gardent 1991; Hardt 1999).

However, the link between natural language and continuations has only recently been made explicit, and this paper's use of control operators for a direct-style analysis is novel.

The analyses presented here are part of a larger project, that of relating computational side effects to *linguistic side effects*. The term "computational side effect" here covers all programming language features where either it is unclear what a denotational semantics should look like, or the "obvious" denotational semantics (such as making each arithmetic expression denote a number) turns out to break referential transparency. A computational side effect of the first kind is jumps to labels; one of the second kind is mutable state. By analogy, I use the term "linguistic side effects" to refer to aspects of natural language where either it is unclear what a denotational semantics should look like, or the "obvious" denotational semantics (such as making each clause denote whether it is true) turns out to break referential transparency. Besides quantification and polarity sensitivity, some examples are:

(64) a. Bob *thinks* Alice likes CS187. (Intensionality)
b. *A man* walks. *He* whistles. (Variable binding)
c. *Which* star did Alice see? (Interrogatives)
d. Alice *only* saw VENUS. (Focus)
e. *The king of France* whistles. (Presuppositions)

To study linguistic side effects, I propose to draw an analogy between them and computational side effects. Just as computer scientists want to express all computational side effects in a uniform and modular framework and study how control interacts with mutable state (Felleisen and Hieb 1992), linguists want to investigate properties common to all linguistic side effects and study how quantification interacts with variable binding. Furthermore, just as computer scientists want to relate operational notions like evaluation order and parameter passing to denotational models like continuations and monads, linguists want to relate the dynamics of information in language processing to the static definition of a language as a generative device. Whether this analogy yields a linguistic theory that is empirically adequate is an open scientific question that I find attractive to pursue.

8. ACKNOWLEDGMENTS

Thanks to Stuart Shieber, Chris Barker, Raffaella Bernardi, Barbara Grosz, Pauline Jacobson, Aravind Joshi, William Ladusaw, Fernando Pereira, Avi Pfeffer, Chris Potts, Norman Ramsey, Dylan Thurston, Yoad Winter, and anonymous referees. This work is supported by the United States National Science Foundation Grants IRI-9712068 and BCS-0236592.

9. REFERENCES

Barker, Chris. 2000. Notes on higher-order continuations. Manuscript, University of California, San Diego.

———. 2002. Continuations and the nature of quantification. *Natural Language Semantics* 10(3):211–242.

Barwise, Jon, and Robin Cooper. 1981. Generalized quantifiers and natural language. *Linguistics and Philosophy* 4: 159–219.

Bernardi, Raffaella. 2002. Reasoning with polarity in categorial type logic. Ph.D. thesis, Utrecht Institute of Linguistics (OTS), Utrecht University.

Bernardi, Raffaella, and Richard Moot. 2001. Generalized quantifiers in declarative and interrogative sentences. *Journal of Language and Computation* 1(3):1–19.

Bernardi, Raffaella, and Øystein Nilsen. 2001. Polarity items in type logical grammar: Connection with DMG. Slides for talk at Learning Logic and Grammar workshop, Amsterdam.

Carpenter, Bob. 1997. *Type-logical semantics*. Cambridge: MIT Press.

[3] Although this paper uses Danvy and Filinski's control hierarchy, polarity sensitivity can be expressed equally well in Shan and Barker's system.

[4] The syntactic distinction among the types Bool, BoolPos, and BoolNeg may even be semantically interpretable via the formulas-as-types correspondence, but the potential for such a connection has only been briefly explored (Bernardi and Nilsen 2001) and we do not examine it here. In this connection, Krifka (1995) and others have proposed on pragmatic grounds that determiners like *any* are negative polarity items because they indicate extreme points on a scale.

Danvy, Olivier, and Andrzej Filinski. 1989. A functional abstraction of typed contexts. Tech. Rep. 89/12, DIKU, University of Copenhagen, Denmark. http://www.daimi.au.dk/~danvy/Papers/fatc.ps.gz.

———. 1990. Abstracting control. In *Proceedings of the 1990 ACM conference on Lisp and functional programming*, 151–160. New York: ACM Press.

Danvy, Olivier, and John Hatcliff. 1992. CPS-transformation after strictness analysis. *ACM Letters on Programming Languages and Systems* 1(3):195–212.

———. 1994. On the transformation between direct and continuation semantics. In *Mathematical foundations of programming semantics: 9th international conference (1993)*, ed. Stephen D. Brookes, Michael G. Main, Austin Melton, Michael W. Mislove, and David A. Schmidt, 627–648. Lecture Notes in Computer Science 802, Berlin: Springer-Verlag.

van Eijck, Jan. 1998. Programming with dynamic predicate logic. Report INS-R9810, Centrum voor Wiskunde en Informatica, Amsterdam. Also as Research Report CT-1998-06, Institute for Logic, Language and Computation, Universiteit van Amsterdam.

van Eijck, Jan, and Nissim Francez. 1995. Verb-phrase ellipsis in dynamic semantics. In *Applied logic: How, what, and why: Logical approaches to natural language*, ed. László Pólos and Michael Masuch. Dordrecht: Kluwer.

Felleisen, Matthias. 1988. The theory and practice of first-class prompts. In *POPL '88: Conference record of the annual ACM symposium on principles of programming languages*, 180–190. New York: ACM Press.

Felleisen, Matthias, and Robert Hieb. 1992. The revised report on the syntactic theories of sequential control and state. *Theoretical Computer Science* 103(2):235–271.

Fry, John. 1999. Proof nets and negative polarity licensing. In *Semantics and syntax in lexical functional grammar: The resource logic approach*, ed. Mary Dalrymple, chap. 3, 91–116. Cambridge: MIT Press.

Gardent, Claire. 1991. Dynamic semantics and VP-ellipsis. In *Logics in AI: European workshop JELIA '90*, ed. Jan van Eijck, 251–266. Lecture Notes in Artificial Intelligence 478, Berlin: Springer-Verlag.

Groenendijk, Jeroen, and Martin Stokhof. 1991. Dynamic predicate logic. *Linguistics and Philosophy* 14(1):39–100.

de Groote, Philippe. 2001. Type raising, continuations, and classical logic. In *Proceedings of the 13th Amsterdam Colloquium*, ed. Robert van Rooy and Martin Stokhof, 97–101. Institute for Logic, Language and Computation, Universiteit van Amsterdam.

Hardt, Daniel. 1999. Dynamic interpretation of verb phrase ellipsis. *Linguistics and Philosophy* 22(2):185–219.

Hendriks, Herman. 1993. Studied flexibility: Categories and types in syntax and semantics. Ph.D. thesis, Institute for Logic, Language and Computation, Universiteit van Amsterdam.

Hobbs, Jerry R., and Stanley J. Rosenschein. 1978. Making computational sense of Montague's intensional logic. *Artificial Intelligence* 9:287–306.

Hobbs, Jerry R., and Stuart M. Shieber. 1987. An algorithm for generating quantifier scopings. *Computational Linguistics* 13(1–2):47–63.

Hung, Hing-Kai, and Jeffery I. Zucker. 1991. Semantics of pointers, referencing and dereferencing with intensional logic. In *LICS '91: Proceedings of the 6th symposium on logic in computer science*, 127–136. Washington, DC: IEEE Computer Society Press.

Kelsey, Richard, William Clinger, Jonathan Rees, et al. 1998. Revised^5 report on the algorithmic language Scheme. *Higher-Order and Symbolic Computation* 11(1): 7–105. Also as *ACM SIGPLAN Notices* 33(9):26–76.

Krifka, Manfred. 1995. The semantics and pragmatics of polarity items. *Linguistic Analysis* 25:209–257.

Ladusaw, William A. 1979. Polarity sensitivity as inherent scope relations. Ph.D. thesis, Department of Linguistics, University of Massachusetts. Reprinted by New York: Garland, 1980.

Lewin, Ian. 1990. A quantifier scoping algorithm without a free variable constraint. In *COLING '90: Proceedings of the 13th international conference on computational linguistics*, vol. 3, 190–194.

May, Robert. 1985. *Logical form: Its structure and derivation.* Cambridge: MIT Press.

Montague, Richard. 1974. The proper treatment of quantification in ordinary English. In *Formal philosophy: Selected papers of Richard Montague*, ed. Richmond Thomason, 247–270. New Haven: Yale University Press.

Moran, Douglas B. 1988. Quantifier scoping in the SRI core language engine. In *Proceedings of the 26th annual meeting of the Association for Computational Linguistics*, 33–40. Somerset, NJ: Association for Computational Linguistics.

Nielsen, Lasse R. 2001. A selective CPS transformation. In *Proceedings of MFPS 2001: 17th conference on the mathematical foundations of programming semantics*, ed. Stephen Brooks and Michael Mislove. Electronic Notes in Theoretical Computer Science 45, Amsterdam: Elsevier Science.

Shan, Chung-chieh. 2003. Quantifier strengths predict scopal possibilities of Mandarin Chinese *wh*-indefinites. Draft manuscript, Harvard University; http://www.eecs.harvard.edu/~ccshan/mandarin/.

Shan, Chung-chieh, and Chris Barker. 2003. Explaining crossover and superiority as left-to-right evaluation. Draft manuscript, Harvard University and University of California, San Diego; http://semanticsarchive.net/Archive/TBjZDQ3Z/.

Taha, Walid, and Michael Florentin Nielsen. 2003. Environment classifiers. In *POPL '03: Conference record of the annual ACM symposium on principles of programming languages*, 26–37. New York: ACM Press.

Thielecke, Hayo. 2003. From control effects to typed continuation passing. In *POPL '03: Conference record of the annual ACM symposium on principles of programming languages*, 139–149. New York: ACM Press.

Parasitic Scope

Chris Barker, *UCSD*

The availability of an internal reading for *same* (likewise *different*) can depend on the presence of a scope-taking element elsewhere in the clause.

(1) John read the same book.

(2) Everyone read the same book.

(1) has only a deictic reading, on which there must be some pragmatically salient book. (2) has in addition a (sentence-)INTERNAL reading on which the book each person read is compared only to the book read by each of the other people.

Keenan (1992, *L&P*) proves that the internal reading cannot be expressed as the compositional interaction of any set of generalized quantifiers. A popular alternative strategy (Stump 1982 ms.; Moltmann 1992 *L&P*; van Eijck 2004 submitted) is to allow noun phrases to form discontinuous (i.e., non-compositional) higher-order ("n-ary") quantifiers.

However, it is not clear how to extend such accounts to cases in which the NP containing *same* is inside another NP:

(3) [Two men with the same name] walked into the room.

Although never noted in the literature (as far as I know), the bracketed noun phrase also has an internal reading, on which a use of (3) will be true just in case any two men who walked into the room have the same name as each other. The challenge for the approaches just mentioned is that the dependent noun phrase *the same name* is contained within the noun phrase with which it would need to combine to form an n-ary quantifier.

I will propose a compositional analysis of NP-internal cases like (3), then show how the proposal generalizes to handle examples such as (1) and (2).

Familiar quantifiers such as *everyone* function syntactically as noun phrases, take scope over a clause, and return a clause as a result, and therefore have semantic type $\langle\langle \mathtt{e}, \mathtt{t}\rangle, \mathtt{t}\rangle$. Analogously, I propose that *same* functions syntactically as an adjective (i.e., a nominal modifier), takes scope over a nominal, and returns a nominal: type $\langle\langle Adj, N\rangle, N\rangle$, or in somewhat more detail, $\langle\langle\langle\langle \mathtt{e}, \mathtt{t}\rangle, \langle \mathtt{e}, \mathtt{t}\rangle\rangle, \langle \mathtt{e}, \mathtt{t}\rangle\rangle, \langle \mathtt{e}, \mathtt{t}\rangle\rangle$.

In fact, this is a fairly natural semantic type for an adjective to have: just as a LIFT type-shifting operation (Partee and Rooth, Steedman, Jacobson, etc.) characterizes the semantic relationship between a non-quantificational noun phrase such as *John* and its quantificational equivalent (namely, the generalized quantifier $\lambda P.P\mathbf{j}$), I show how LIFT also characterizes the relationship between a non-quantificational adjective such as *tall* and the quantificational type proposed for *same*.

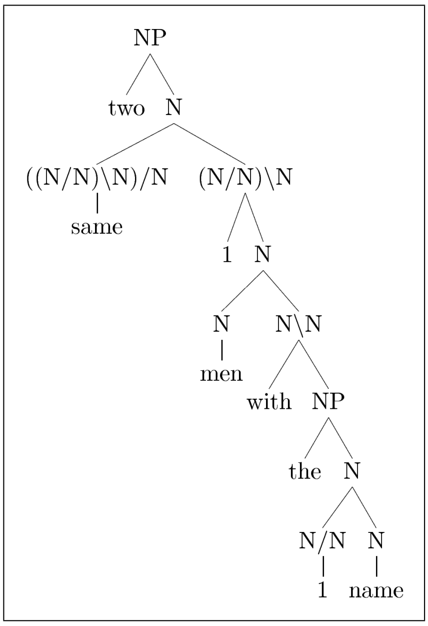


Figure 1: *two men with the same name*, after QR'ing *same*

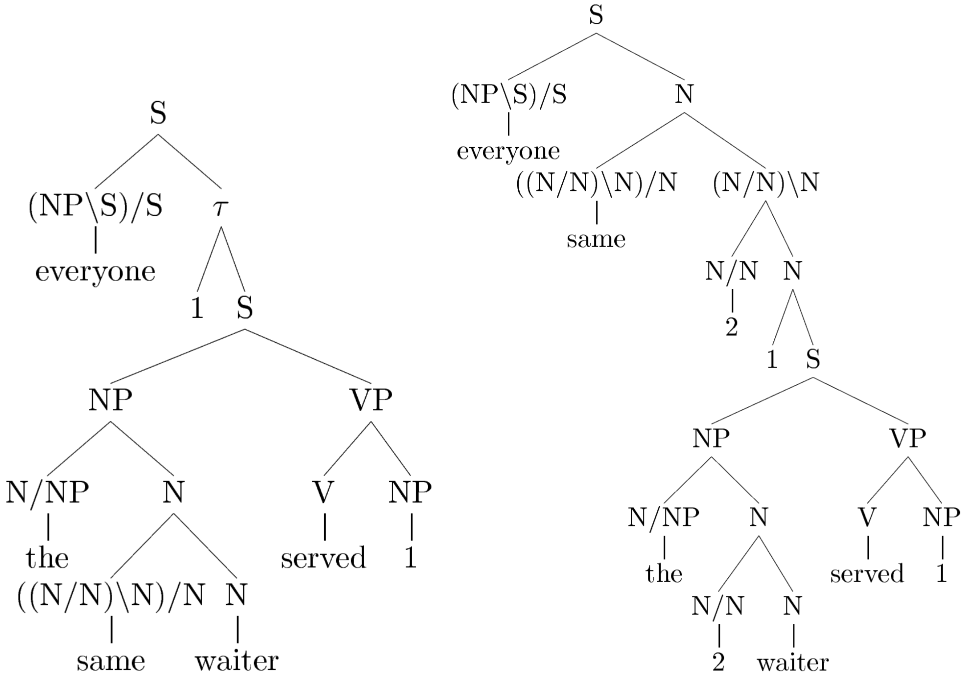
(4) a. $[\![\textit{same}]\!] = \lambda F.\lambda X.\exists A \in \mathbf{choice} : \forall c \in \mathbf{Cov}(X) : F(A)(c)$

b. $[\![\textit{men with the same name}]\!] = \lambda X.\exists A \forall c \in \mathbf{Cov}(X) : \mathbf{with}(\mathbf{the}(A(\mathbf{name})))(c)$

Given (4a), *men with the same name* will denote the property that is true of a set of entities X just in case there is some choice function A (an adjective meaning mapping each nominal to a singleton set), and every member of the pragmatically-supplied cover $\mathbf{Cov}(X)$ has the (unique) name picked out by $A(\mathbf{name})$. Distributing over the elements of a cover builds in the distributivity that Carlson (1985 *L&P*) argues is essential to *same*, but unlike Carlson's proposal, does not make direct reference to any set of events (which is a good thing, in view of NP-internal uses like (3)!).

The same syntactic and semantic analysis easily accounts for (5) (and similarly for (2)):

(5) The same waiter served everyone. [Stump, Heim]



We derive parasitic scope as follows. First step (the leftmost tree): quantifier-raise *everyone* Heim-and-Kratzer style (1998:186), i.e., with an unnamed node τ dominating the QR index (in this case, '1'). Note that the semantic type of τ is $\langle \mathtt{e}, \mathtt{t} \rangle$, the same semantic type as a nominal. Second step: allow *same* to take scope at τ (or even simpler, choose τ = N), in which case *same* hijacks the nuclear scope of *everyone*. Because *same* doesn't even have a suitable place to take scope until after *everyone* has undergone QR, the scope of *same* is parasitic on that of *everyone*. Then using again (4a), the truth conditions require that (5) will be true only if there is a choice function A such that everyone was served by the waiter picked out by $A(\mathbf{waiter})$.

I will also present the analysis in a non-derivational, direct compositional, variable-free, continuation-based framework. That formulation allows a further simple generalization that accounts (for the first time) for examples in which the trigger is not a plural or quantificational NP, e.g., the internal reading of *John read the same book quickly and slowly*.

DRAFT

Parasitic Scope

Chris Barker, *UCSD*

> [*Same* and *different*] *appear to be totally resistant to a strictly compositional semantic analysis...* Stump (1982:2)

Abstract: Keenan (1992) proves there is no generalized quantifier that expresses the meaning *the same two books*. And indeed, previous analyses of adjectives like *same* have been heavily pragmatic (Dowty 1985, Beck 2000) or else deliberately non-compositional, either building syntactically discontinuous higher-order quantifiers (Keenan 1992, van Eijck 2003), or relying on side calculations (Stump 1982, Moltmann 1992). Building on insights of Carlson (1987), I propose the first strictly compositional semantic account of *same*. New data, including especially NP-internal uses such as *two men with the same name*, suggests that *same* is a quantificational element taking scope over nominals. Given lift as a basic type-shifting operator, I show that this proposal follows naturally from the fact that *same* is an adjective. Independently-motivated assumptions extend the analysis to standard examples such as *Anna and Bill read the same book* via a mechanism I call parasitic scope, in which the scope of *same* depends on the scope of some other scope-taking element in the sentence. Although I initially express the main analysis within a movement-based framework with Quantifier Raising in the style of Heim and Kratzer (1998), I go on to implement the analysis within a continuation-based, variable-free, directly compositional combinatory-categorial grammar, without Quantifier Movement or any use of Logical Form distinct from surface structure. The empirical payoff for dealing in continuations is that a simple generalization accounts for the first time for cases in which *same* distributes over objects other than NP denotations, as in the relevant interpretation of *John hit and killed the same man.*

This is a rough draft of July 2, 2004 based on my SALT 14 talk at Northwestern University on 15 May 2004. Please check my web page (`http://ling.ucsd.edu/~barker`) for revised versions!

1. A compositional semantic account of *same*

This paper seeks to understand the semantic behavior of *same* (with more limited discussions of some related adjectives, notably *different*) in a variety of its most typical uses, including (1):

(1) Anna and Bill read the same book.

There is a deictic reading of (1) that depends on identifying some contextually-salient book (see section 1.1). But what will be the central topic of this paper is a distinct interpretation on which a use of (1) will be true just in case there exists some book x—any book x—such that Anna read x and Bill read x. Carlson (1987) calls this second type of interpretation an INTERNAL reading, which he describes (p. 532) as a case in which "the sentence, in some way or other, provides its own context".

Remarkably, as far as I know, there has never been a compositional semantic analysis of the internal reading of sentences like (1).

This is not because *same* is in any way exotic or rare; quite the contrary, *same* is deeply ingrained into the core of English, across all registers and dialects; furthermore, similar remarks apply to analogous expressions in other languages.

Nor is there any paralyzing uncertainty about what the truth conditions are, at least for relatively simple examples like (1). Setting aside the vagueness inherent in deciding how similar two objects have to be in order to count as the same (see section 1.2), the truth conditions of (1) are clear and robust. Indeed, the stark difference in meaning between (1) and the same sentence with *same* removed (i.e., *Anna and Bill read the book*, or *a book*) is exactly the sort of meaning difference that semanticists are usually most eager to analyze.

Perhaps one reason that a compositional semantic treatment has not been proposed in almost a quarter-century of study is that standard semantic techniques seem to lead to a dead end (witness the quotation from Stump (1982) above). Indeed, as discussed below in section 2, Keenan (astonishingly!) proves that there is no composition of generalized quantifiers that can possibly express the internal meaning of (1).

Yet despair would be premature. Keenan's result does not mean that *same* is non-compositional; all it shows is that the meaning of *same* cannot be expressed in terms of generalized quantifiers. But after all, *same* is an adjective, not a determiner or an NP. I suggest below that the characteristic behavior of *same* falls out once we recognize that it is a scope-taking adjective, and not an NP. In fact, I argue in section 5 that the existence of such scope-taking adjectives arises quite naturally in any system that recognizes LIFT as a legitimate type-shifting operation (in the presence of a sufficiently general theory of scope-taking).

As emphasized by Carlson (1987), the conditions under which *same* takes scope often depend on the presence of other scope-taking elements elsewhere in the sentence.

(2) a. The same waiter served John.
b. The same waiter served everyone. [Stump, Heim]

The standard judgment (with which I agree) is that (2b) has an internal interpretation that (2a) lacks, and the availability of the extra reading has to do with the fact that *everyone* is quantificational. I will argue that the scope of *same* depends on the scope of *everyone* in a certain way that I will call PARASITIC SCOPE, for reasons that will become clear in section 6.

I should hasten to say that even though *same* has not received an analysis that is both compositional and semantic, it has received insightful analyses that are either compositional or semantic. Among the compositional analyses are Dowty (1985) and also Beck's (2000) analysis of *different*, both of which rely heavily on pragmatically-controlled free variables. Relying on free variables simplifies the combinatorics, at the cost of denying that there is any formal link between, for instance, the denotation of *Anna and Bill* in (1) and the properties that pick out the book in question on the internal reading.

Of course, whether an analysis ought to be semantic (i.e., combinatoric) or else pragmatic (in this case, relying on free variables) is legitimately debatable. I provide arguments below in section 3 that pragmatic approaches have empirical shortcomings compared to the semantic approach developed below.

There are also explicit formal analyses that are semantic (combinatoric) but not compositional, including Stump's (1982) pioneering treatment, Moltmann (1992), and van Eijck (2003). These analyses either allow for side calculations carried out in parallel with normal composition (Stump, Moltmann), or else combine discontinuous NPs into higher-order ("n-ary") quantifiers, i.e., treat *Anna and Bill ... the same book* as a semantic unit, as in van Eijck, building on suggestions in Keenan (1992). As each of these authors point out, the reason these analyses fail to be compositional is that they require combining elements semantically in an order that is incompatible with gross syntactic constituency.

Now, whether a particular phenomenon ought to be treated compositionally is also open to debate, though the decision may ultimately depend on methodological preference. But it is one thing to resort to a non-compositional analysis when it is the only type of analysis available, and quite another to do so when a compositional analysis exists. So one important goal of this paper is to show that a strictly compositional semantic analysis is not only possible, but fairly elegant.

Finally, I should also note that there are insightful discussions of *same* and *different* that assume that a compositional semantic analysis is possible, but do not provide complete details, notably Carlson (1987; see especially remarks on pages 531, 541, 545). In some sense, then, the analysis below justifies Carlson's optimism that a compositional semantic account exists. As may already be clear, I have relied heavily on Carlson's insights at every stage in the research reported here, especially concerning the relationship between distributivity and adjectives like *same*. However, the analysis below departs from some of Carlson's main hypotheses in other ways; in particular, the NP-internal uses of *same* discussed below (e.g., *two men with the same name*) show that *same* does not require any direct reference to events, one of Carlson's main conclusions.

Although I initially develop the analysis below in terms of quantifier raising at LF in the style of Heim and Kratzer (1998), in part for the sake of expository familiarity, the analysis does not depend in any essential way on movement, quantifier raising, or LF. I demonstrate this by translating the analysis into a continuation-based, variable-free, directly compositional combinatory categorial grammar. The empirical payoff for switching to a continuation-based treatment is the first-ever formal account of examples in which *same* distributes over non-NP denotations, as in the relevant reading of *John hit and killed the same man*.

I believe that it is no accident that the first compositional semantic account of *same* falls out from a continuation-based approach. Continuations are a technique originally developed for studying the semantics of programming languages. I will not devote much space below to motivating or characterizing continuations (see Barker (2002), de Groote (2001), Shan and Barker (2003), Barker and Shan (2004), and references there), concentrating instead on explaining the behavior of *same*. Nevertheless, one of my main motivations for studying *same* is to provide support for the claim that continuations provide new and valuable insights into the nature of scope-taking.

I conclude that despite Stump's pessimism and Keenan's discouraging result, *same* does in fact have a perfectly reasonable strictly compositional semantic treatment, and that the discovery of such an analysis supports the claim that continuations are ideally suited for reasoning about scope-taking.

1.1. First preliminary: deictic same. *Same* always has a deictic use that depends on identifying some salient object present in the discourse context. For instance, if Ivan holds up a copy of Jane Austen's *Emma*, Jorge might utter (8):

(3) Hey, I just read the same book!

In the described situation, it is appropriate to assume that *same* conveys the property that an object has if it is held by Ivan. And in fact, this kind of context-dependent reading is the only interpretation *same* has in (3).

But if there is a plural NP in the sentence, another interpretation emerges:

(4) a. Anna and Bill read the same book. (same as (1))
b. Anna and Bill read the held-by-Ivan book. **Deictic**
c. Anna and Bill read the read-by-Anna-and-Bill book. **Internal**

In the same situation described for (3), (4a) can certainly have a deictic use as paraphrased in (4b) that asserts that Anna and Bill read the book that Ivan is holding. But (as discussed above) there is another, "internal", interpretation, as paraphrased in (4c) on which (4a) is true whenever there is some book that Anna and Bill each read.

Like most authors (though by no means all, e.g., Dowty (1985)), I will assume that the internal reading is a systematic interpretation distinct from the deictic uses, and that it requires a formal grammatical account. I will argue explicitly against attempting to unify the internal and deictic uses in section 3.

Carlson notes that one of the key differences between the deictic use and the internal use is that an internal use is only possible if the trigger NP is interpreted distributively. That is, the interpretation in (4b) is consistent with Anna and Bill reading the book collaboratively, but the internal reading in (4c) entails that Anna and Bill each read the book independently. As Carlson also noted, although there must always be some element for internal *same* to distribute over, it need not be an NP meaning. Such uses with non-NP triggers are discussed in section 4 and analyzed in section 7.

1.2. Second preliminary: Types versus tokens. One fascinating aspect of the semantics of *same* (and similar expressions such as *different*, *opposite*, etc.) is variation in just how similar two objects have to be in order to count as the same (or how different, etc.).

(5) I drive a Ford Falcon and Enzo drives the same car.

For instance, as Nunberg (1984) notes, (5) can be true even though the speaker and Enzo drive different objects, as long as both cars have the same make and model. In other words, the cars need only be type-identical, not token-identical. Lasersohn (2000), discussing Nunberg's account, persuasively argues that the difference between type identity and token identity is a different in degree, not in kind. In any case, I will assume that this phenomenon is orthogonal to the compositional issues discussed here, and will play no further part in this paper.

2. Beyond the Frege boundary

Keenan (1992) proves there is no set of generalized quantifiers that can be composed to adequately render the contribution to truth conditions of (6a).

(6) Anna and Bill read the same two books.

To see the form of the proof, it is necessary to view the sequence of NPs *Anna and Bill ... the same two books* as a discontinuous predicate over relations. For instance, in (6a), this sequence combines with the transitive verb *read* to form a complete sentence. In Keenan's terminology, a sequence is REDUCIBLE if it can be decomposed into separate generalized quantifiers that accurately reflect the truth conditions of the orginal complex expression. So the main question here is whether the sequence *Anna and Bill ... the same two books* is reducible. If not, then according to Keenan, that sequences lies "beyond the Frege boundary".

Keenan answers this question by proving Reducibility Equivalence: if two sequences are reducible, and if they yield the same truth value whenever the transitive verb meaning happens to be a cross product, then the two sequences must be completely equivalent, i.e., give the same result for every transitive verb meaning.

(7)a. Anna and Bill read the same two books.
b. (Both) Anna and Bill read exactly two books.

In order to prove that NPs containing *same* are not reducible, Keenan's strategy is as follows: we first establish that the sequences *Anna and Bill ... the same two books* and *Both Anna and Bill ... exactly two books* yield the same truth value for any cross-product relation, and then we observe that they give different truth values on at least one other (non-cross-product) relation.

The first step is to establish that the two sequences give the same result for any cross-product relation. If *read* denotes a cross product, then every reader in the domain reads every book. In any situation in which there are more than two books, both (6a) and (6b) will be false. This is because when *read* denotes a cross product, Anna and Bill will each have read every book in the domain, and if there are more than two books, that will falsify both sentences. Similarly, in any situation in which there are fewer than two books, both sentences will be false. Therefore, assume that there are exactly two relevant books. Because *read* is a cross product, Anna and Bill (or however many people end up in subject position) each read both books; this is sufficient to satisfy the truth conditions of both sentences. Thus (6a) and (6b) are defined (felicitous) over the same set of models, and they yield the same truth value for any relation that is a cross product.

By Reducibility Equivalence, if the sequence *Anna and Bill ... the same two books* is reducible, then (6a) and (6b) must be synonymous. The next step, then, is to show that there is at least one possible (non cross-product) relation for which the two sentences are not synonymous. Therefore imagine that Anna reads exactly two books, and that Bill also reads exactly two books, but the books Anna reads are different than the books that Bill reads. Then (6a) is false (they didn't read the same books), but (6b) is true (they read exactly two books each). Assuming that the NPs *Anna and Bill*, *both Anna and Bill*, and *exactly two books* can be rendered by garden variety generalized quantifiers along the lines proposed in Barwise and Cooper (1981), we can deduce that *the same two books* cannot be adequately translated by any generalized quantifier, and furthermore that the culprit must be the presence of *same*.

A proof of Reducibility Equivalence and additional details concerning this specific example can be found both in Keenan (1992) and also in Dekker (2003); see also van Eijck (2004) for additional results concerning reducibility.

How could there be any compositional analysis given Keenan's result? One possible answer is that we could recognize the existence of discontinuous (i.e., non-compositional) quantifiers such as *Anna and Bill ... the same two books*. This is in effect the proposal of Stump (1982). As the analysis of a sentence proceeds, Stump places each NP onto a Cooper store. Certain types of NP, including NPs containing *same*, are able to interact with other NPs while in the store, in effect forming a discontinuous constituent. Later, the sequence of NPs can be cashed out and applied to a transitive verb meaning. Van Eijck (2003), building on suggestions of Keenan (1992) proposes a similar strategy.

Discontinuous quantifier strategies are perfectly coherent and precise, and they are weakly compositional in the sense that the meaning of the whole depends on the meanings of the parts. But they are certainly not directly compositional in the sense of, e.g., Jacobson (1999). Direct compositionality is a particularly strict form of compositionality on which each syntactic constituent has a denotation that depends only on the meanings of its immediate subconstituents. So unless it is possible to justify *Anna and Bill ... the same two books* as a syntactic constituent (which seems unlikely), we must conclude that the analyses of Stump and van Eijck fail to be compositional.

A second possible answer, and the one pursued here, is that Keenan's result only bears on the possibility of reducing NP meanings to (what he calls type <1>) generalized quantifiers. If we allow NPs to denote objects other than generalized quantifiers, Keenan's result does not apply. I will argue below that although *same* does take scope, it is not a generalized quantifier. But this should not be surprising: after all, *same* is an adjective, not an NP! Put another way, Keenan's result is only a show-stopper if we assume that the only kind of scope-taking expression is an NP.

3. The internal reading is not a special case of the deictic reading

Given that a deictic reading is always available for *same*, one obvious and important question is whether the internal reading could be adequately treated as a special case of a deictic reading. After all, pronouns standardly receive exactly this kind of analysis:

(8) a. Mary saw him$_i$.
b. Everyone$_i$ thinks she$_{i,j}$ is intelligent.

On the traditional analysis, pronouns translate as variables. (8a) has only a deictic reading on which the pronoun translates as a variable that receives its value from context. (8b) has both a deictic and a quantificationally-bound reading (potentially analogous to an internal reading for *same*), depending on whether the pronoun translates as some independent variable (i.e., 'j'), or translates as the same variable bound by the quantifier introduced by *everyone* (i.e., 'i'). There is no difference in the analysis of the pronoun; the bound reading arises when the variable that serves as the translation of the pronoun happens to coincide with the index of some other element in the same sentence.

Perhaps, then, *same* introduces some variable, and the internal reading arises when that hypothetical variable is bound by some other element in the sentence.

(9) Two women in this room have the same name.

On the deictic reading, the speaker may have a specific name in mind, as when (9) is used in the following monologue: "My friend Heddy's name is highly unusual; nevertheless, two women in this room have the same name." In such a context, (9) will be true only if there are two women in the room whose name is *Heddy*.

On the internal reading, in contrast, (9) will be true just in case there is any name such that two women in the room have that name. To emphasize the quantificational nature of these truth conditions, note that a speaker might assert (9) on the basis of a mistaken belief that there are two women in the room named *Heddy*. But if, unbeknownst to the speaker, there are two women named *Mary* in the room, the sentence is nevertheless true, albeit accidentally.

In view of these observations, it seems inescapable that on the internal reading, some element in the sentence must in effect introduce an existential quantifier over names. Nor can we pin the existential force on the cardinal *two*:

(10) Everyone read the same book.

In addition to the deictic reading (on which we have a specific book in mind, say, *Emma*), (10) has an internal reading on which it is true if there is any book such that everyone read that book.

Dowty (1985) proposes an analysis of *same* that explicitly attempts to reduce the internal reading to a special case of the deictic reading. On Dowty's proposal, in addition to introducing existential quantification, the denotation of *same* also introduces two contextually-determined variables, C and R.

(11) $[\![\mathit{same}]\!] = \lambda N.\lambda x \exists f : \{x\} = f(N) \wedge \forall c < C : Rxc$ [Dowty]

Here, C is a contextually-specified set of individuals that Dowty calls a comparison class. In the case of (10), C will be assigned by the context to the set of individuals quantified over by *everyone*. The relation R is a contextually-specified relation over individuals, the relation that holds between each member of the comparison class and the object x that the predicate *same book* picks out. In (10), R will be $[\![read]\!]$, so (10) will assert that there is some book x such that everyone read x.[1]

Beck (2000) proposes an analysis for *different* that also crucially relies on a contextually-supplied comparison set (though it is not clear whether her analysis generalizes to *same*). Analogously to Beck's discussion of *different*, there is a parallel between the uses of *same* here with uses that have an overt *as* phrases (e.g., *the same book as everyone else read*), which arguably motivates reference to a comparison class and a comparison relation.

Bearing in mind that one of Dowty's goals is to unify the deictic and the internal readings, certainly there will always be a choices for C and R that result in appropriate truth conditions for the deictic reading. Dowty provides an example involving *different* motivating the claim that the values of R and C can be determined from outside the sentence (i.e., deictically):

(12) The teachers discussed *Taxi Driver*, but the students saw a different movie.

If we choose $C =$ the teachers and $R =$ discussed, we get appropriate truth conditions for (12): each student saw a movie that is different from a discussed movie.

However, it is not clear that there is ever a situation in which a deictic use is able to exploit the full truth-conditional power provided by access to a comparison set.

(13) The men discussed a house. John read the same book.

If we could choose C as the set of men and R as the relation **discussed**, (13) would assert that there exists a book such that each of the men discussed that book, and that is the book that John read. But there is no such reading; nor is there even a reading on which John read the same book that each of the men read (i.e., choosing $C =$ the men and $R =$ read).

Now, the absence of logically possible readings can perhaps be explained on pragmatic grounds (though it is not obvious to me what such an explanation would look like in this case). But there seems to be a systematic pattern at work: whenever one of C or R takes its value from the same clause as *same*, the other does too.

(14) Anna and Bill read the same book.

If the comparison class C is the set consisting of Anna and Bill, then the relation R must be **read**.

Even more suspiciously, there seems to also be a systematic relation between the values of R and C for the internal case: on any internal reading, R always turns

[1] I have taken some liberties in my presentation of Dowty's proposal in an order to facilitate comparison with my own proposal below. Most notably, the existential quantifier in (11) quantifies over adjective meanings, though Dowty's version quantifies over individuals (leading to equivalent truth conditions). In any case, I have (I hope) faithfully preserved the the roles of C and R.

out to be the remainder of the clause after the NP giving rise to the comparison class C has been subtracted.

(15) Anna and Bill must have read the same book.

For instance, there is a reading of (15) on which C is the set consisting of Anna and Bill and R is the remainder relation **must-have-read**; but there is no reading on which C is Anna and Bill but R is just **read** (in which case (15) would express a tautology).

What I'm suggesting, then, is that there is a systematic correlation between the choice of C and the choice of R that goes unexplained on the pragmatic account. As Carlson (1987:532) puts it, on the internal reading, "the comparison is somehow made available by virtue of the meaning of the sentence itself". We shall see that on the semantic account below, at least for the internal reading, the relation that serves the role of R systematically corresponds to what is left over after subtracting the NP corresponding to the comparison set.

There are also some empirical difficulties for the unified analysis, at least in the version of the analysis given in (11).[2]

(16) a. The men or the women read the same book.
b. Ann read and Bill reviewed the same book.

In (16a), the truth conditions are clear: either the men read the same book, or else the women read the same book. Yet there is no choice for C that gives the correct truth conditions.

Similarly, for the right node raising example in (16b), there is no suitable choice for R. In particular, we can't chose the complex relation of reading-or-reviewing, since there would be no way of guaranteeing that what Ann did to the book was read it, or that what Bill did to the book was review it.[3]

In any case, I will take the discussion in this section as motivating at least considering a semantic (combinatoric) treatment.

4. Distributivity, events, and NP-internal *same*

Carlson (1987) makes a strong case that the availability of an internal reading depends on distributing over events:

(17) John read the same book.

In (17), there is just a single reading event, and only a deictic reading is possible.

[2] Dowty's paper, though highly insightful, remains unpublished in draft form; a more developed analysis might very well have anticipated examples like those discussed here.

[3] Needless to say, these examples only come out in favor of a compositional semantic account if I can show how they would be dealt with in a precise and explicit way. I have not included derivations of these examples in this version of this paper, but would be happy to furnish details upon request (my email address is barker@ucsd.edu). The treatment of disjunction follows the proposal in Barker (2002, section 4), and a variant of the Geach rule discussed below in section 7.6 handles Right Node Raising.

(18)a. John and Bill read the same book. Conj. NPs
b. The men read the same book. Plural NP
c. John hit and killed the same man. Conj. Vs
d. John read the same book yesterday and today. Conj. Adv.
e. John read the same book quickly and slowly. Conj. Adv.

In (18), in contrast, internal readings are also available—but only when the sentence describes multiple events. In (18a), for instance, if John and Bill read the book collaboratively, the internal reading disappears. Similarly for (18b), the men in question must each participate in a separate reading event. In (18c), in order for an internal reading to be available, the hitting and the killing must be separate events (i.e., the blow was not in and of itself fatal.) In (18d), if John read slowly and continuously for 48 hours, the internal reading disappears; and finally, in (18e), *quickly* and *slowly* cannot describe the manner in which John read different chapters—rather, they must describe the manner in which John performed separate readings of the book.

One of the most interesting aspects of *same* is the variety of triggers for the internal reading: in (18), we have coordinated NPs, other types of plural NP, coordinated verbs, coordinated adjuncts, and coordinated adverbs. Singular quantificational NPs are also capable of triggering an internal reading:

(19)a. Everyone read the same book.
b. No one read the same book.
c. John read the same book every quarter.
d. John read the same book twice.

As (19d) shows, even a quantificational adverb can trigger an internal reading.

It appears to be a sufficient condition for the availability of an internal reading that some element in the sentence—any element—entails the existence of multiple events.

Yet event multiplication (as Tovena and Van Peteghem (in press) call it) is not a necessary condition for the internal reading.

(20) [Two men with the same name] are sitting in this room.

Although never before noted in the literature (as far as I know), NP-internal uses of *same* such that in (20) also clearly can have an internal reading. On the deictic interpretation, (20) makes a claim about the prevalence of some specific name; on the internal reading, (20) will be true if there is any name such that two men sitting in the room share that name, whether the speaker is aware of what that name might be or not.

If we want to claim that the internal reading requires multiple events, then we must decide that having a name must qualify as an event. I will assume instead that what the internal reading requires is multiple situations. Distinct events certainly count as distinct situations, but configurations of objects described by non-verbal relations can also serve as distinct situations.

5. Why nominal scope is natural for *same*

The general strategy I will take in the remainder of the paper is to begin by analyzing the NP-internal use of *same*. This will give us a basic syntactic and semantic analysis. Then I will extend the analysis to more familiar examples. In each case, the adjustments are fairly minor. NP-internal *same*, then, tells us almost everything we need to know about *same*.

In this section, I show that—given LIFT as a basic type-shifting operation—we can predict the possibility of adjectives behaving the way that *same* behaves. In other words, I claim that it is natural for adjectives to take scope. Furthermore, I claim that when they do, it is natural for them to take scope over nominals.

5.1. NP-internal same is essentially quantificational. The truth conditions for the internal reading entail that the meaning of *same* must be essentially quantificational, since (20) will be true if there is any suitable name. (This is the same conclusion we came to in section 3 with respect to other uses of *same*.) A paraphrase of the internal reading of (20), then, might be

(21) $\exists f_{\texttt{choice}}$. two men with the f name are sitting in this room.

This paraphrase quantifies over adjective meanings (type $\langle\langle \texttt{e}, \texttt{t}\rangle, \langle \texttt{e}, \texttt{t}\rangle\rangle$. But not just any adjective meaning will do; *same* insists on a value that is, in effect, a choice function, in the following sense: given any set of entities X, it will return a singleton set chosen from X. Thus (20) will be true only if there is some choice function f that picks out (a singleton set containing) a specific name (for instance, $f(\mathbf{name})$ might be $\{Ed\}$) such that two men sitting in the room have that name.

Given that there is existential quantification involved, we must try to determine where that existential quantifier can take scope.

(22) [John met two [men with the same name]]
↑ ↑
not here here

For the example in (22), either of the scope possibilities indicated will do. But if *two* is replaced with a determiner that is downward monotonic on its first argument, then allowing the existential to take wide scope gives inappropriately weak truth conditions:

(23) a. John met fewer than three men with the same name.
b. $\exists f$. John met fewer than three men with the f name.

The paraphrase in (23b) will be true if there is any name such that John has met fewer than three men with that name. But those truth conditions are too weak for any reading of (23a): in particular, the internal reading of (23a) should never be true merely because John has met only two people named *Orville* during his life.

Therefore I will assume that the existential quantifier introduced by *same* can take scope at the level of the nominal. There are other logical possibilities for the scoping of the existential introduced by *same* besides the nominal position, of course, but we shall see immediately below that other considerations converge on the nominal as the natural place for the existential to take scope.

5.2. Anatomy of a quantifier. Moortgat (e.g., 1997) generalizes over scope-taking expressions by providing a category constructor q (‘q’ for $q$uantificational) that builds a scope-taking category by combining three elements: a local syntactic Personality, a scope Target, and a final Result category. The category of *everyone*, for instance, is q(NP,S,S): locally, *everyone* behaves as an NP, takes scope over a clause of category S, and produces as a result another clause.

The semantic type of a scope-taking expression q(P,T,R) is $\langle\langle P', T'\rangle, R'\rangle$, where P', T', and R' are the types of categories P, T, and R, respectively. Assuming that the natural basic semantic type of a non-quantificational NP such as *John* is $\texttt{e}$ (the type of an individual), and that the (extensional) type of a clause is $\texttt{t}$ (the type of a truth-value), then *everyone* will have semantic type $\langle\langle \texttt{e}, \texttt{t}\rangle, \texttt{t}\rangle$—naturally, the type of a generalized quantifier.

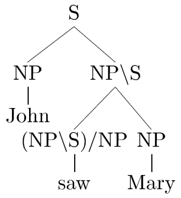
For most scope-taking expressions, the target category and the result category are the same (schematically, q(P,X,X)), though not always. For instance, in-situ wh-phrases might reasonably be analyzed as having category q(NP,S,Q): something that functions locally as an NP, takes scope over a clause, and turns that clause into a question.

I have suggested above that *same* is a scope-taking element that functions locally as an adjective and takes scope over a nominal. Assuming that the result category is the same as the scope target, we can anticipate that the category that we will arrive at below for *same* will be q(Adj, N, N).

5.3. An indispensable type-shifting operator: LIFT. Most grammatical systems that allow any type-shifting at all provide a shifting operation identical to or closely related to an operator that I will call LIFT, including Partee and Rooth (1983), Partee (1987), Hendriks (1993), Jacobson (1999), Steedman (2000), and many others.

I will take for basic category labels NP, S, and N (where N is the category of nominals such as *men*). In order for syntactic categories to record the effect of type-shifting, in addition to the basic categories, we will need structured syntactic categories. Therefore we will also have categories of the form A\B and B/A, where A and B are categories. Then the categories of verbs and verb phrases can be built up from the basic categories as follows:

(24)



As in Lambek (1958), slashes lean in the direction of the expected argument: *saw* expects an NP on its right to form a verb phrase of category NP\S, which in turn expects an NP to its left to form an S. Categories of the form B/A and A\B have semantic type $\langle A', B'\rangle$.

LIFT says that any expression in category A will also be in category B/(A\B). LIFT is so basic and natural that in Lambek’s (1958) grammar, and therefore all type-logical grammars based on Lambek's work, LIFT doesn't even need to be stipulated, since it is a theorem of the logical system:

(25) LIFT as a theorem in the Lambek calculus: $\mathrm{A} \vdash \mathrm{B}/(\mathrm{A}\backslash\mathrm{B})$:

$$\dfrac{\dfrac{\mathrm{A} \quad \overline{\mathrm{A}\backslash\mathrm{B}}^{\,1}}{\mathrm{B}}\,\backslash\mathrm{E}}{\mathrm{B}/(\mathrm{A}\backslash\mathrm{B})}\,/\mathrm{I},1$$

In particular, if the proper name *John* is in the category NP, then *John* is automatically also in the category S/(NP\S) by virtue of LIFT.

Note that the derived category S/(NP\S) has semantic type $\langle\langle \mathtt{e},\mathtt{t}\rangle,\mathtt{t}\rangle$, the semantic type of a generalized quantifier. In other words, the LIFT operation characterizes, among other things, the relationship between individual-denoting NPs like *John* and their generalized-quantifier counterparts. Furthermore, the Curry-Howard labelling of the proof in (25) tells us that the denotation of the generalized quantifier version of *John* will be $\lambda P.P\,\mathbf{j}$, where $\mathbf{j}$ is the individual denoted by *John*, which is exactly the right meaning for the generalized quantifier version of *John* to have.

What this suggests is that we can approximate Moortgat's q operator using slashed categories: an expression with category $q(\mathrm{P},\mathrm{T},\mathrm{R})$, then, corresponds to the slashed category $\mathrm{R}/(\mathrm{P}\backslash\mathrm{T})$.[4]

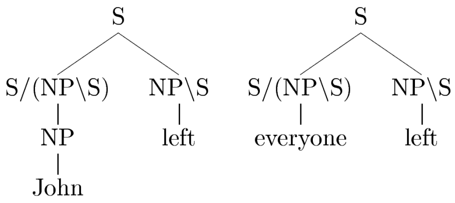
In this instance, then, LIFT turns an expression with no scope-taking ability into a quantificational expression. Analogously, I will show how judicious application of LIFT will turn a normal (non scope-taking) adjective into a scope-taking adjective suitable to serve as the category of *same*.

One way of putting it is that the LIFT operation makes *John* aware of its surroundings. We can gloss the category of a verb phrase NP\S as saying "I need an NP to my left in order to be a complete S". Then the gloss on the category of the LIFTed, version of *John*, S/(NP\S), is "I need something that needs me."

Although LIFT enables every NP to take on the semantic type of a generalized quantifier, only some NPs have meanings that take advantage of the additional expressive range available to the generalized quantifier type. For instance, *everyone* has a meaning that is essentially quantificational, and therefore can only be expressed by a generalized quantifier. This means that *everyone*'s most basic category is S/(NP\S), i.e., a category that looks like a lifted non-quantificational NP.

[4] In section 7, it will be necessary to distinguish between normal slashes, such as the slash in NP\S that governs the linear order of arguments, versus quantificational slashes. In the more detailed system in section 7, $q(\mathrm{P},\mathrm{T},\mathrm{R})$ will be rendered as $\mathrm{R}/\!\!/(\mathrm{P}\backslash\!\!\backslash\mathrm{T})$, where $\backslash\!\!\backslash$ and $/\!\!/$ constitute a mode of combination that is parallel to but distinct from / and \. But it will significantly simplify discussion here to collapse the both modes onto / and \.

(26)

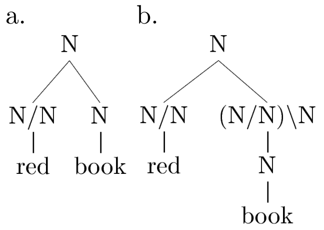


In general, then, scope-taking expressions live at the level of a LIFTed non-scope-taking expression.

As mentioned above, our goal is to arrive at a category for *same* of N/(Adj\N): locally, an adjective, taking scope at a nominal, and producing a nominal as a result. We'll get there the same way we arrived at a generalized quantifier from a non-quantificational NP, i.e., by LIFTing; except that in this case, we'll need to lift twice.

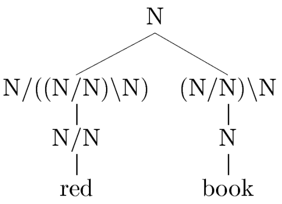
First, we can implement the category Adj as an expression that expects a nominal N to its right to form a complex nominal:

(27)



In our anthropomorphic interpretation, the adjective *red* is the kind of expression that needs an N, and the basic nominal *book* in (27a) satisfies that need. After we lift *book*, as in (27b), the adjective is still seeking an N, but the nominal has had its consciousness raised, so that now it is seeking something that seeks a nominal: "I need something that needs me."

(28)



Now we can lift the adjective, so that it says "I need something that needs me to need it".

At this point, we have the category we are seeking, which is q(N/N, N, N) = N/((N/N)\N): locally, an adjective, taking scope over a nominal.

Two observations: although we have arrived at this point in steps via two applications of LIFT, the relationship between the basic Adjective category N\N and the scope-taking category N/((N/N)\N) is characterized by a single application of LIFT. Second, the Curry-Howard labelling of the LIFTing operation provides a denotation for the lifted adjective that parallels the generalized quantifier denotation of a proper name: $\lambda P.P\,\mathbf{red}$, where **red** is the basic (i.e., category N/N) meaning of *red*.

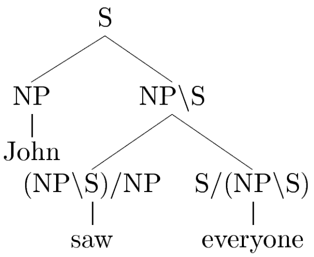
Just as a quantificational NP such as *everyone* has the category of a LIFTed non-quantificational NP such as *John*, a scope-taking adjective such as *same* has the category of a LIFTed normal adjective like *red*.

5.4. From type clash to Quantifier Raising, Heim and Kratzer style. Now that we have a quantificational category, we need to provide a mechanism that will allow it to take scope. I will develop my initial discussion of scope in the style of Heim and Kratzer's (1998) textbook for several reasons, not least of all because it is simple, well motivated, and familiar to most readers.

In addition, Heim and Kratzer's specific implementation of quantifier raising has one unusual detail that will turn out to be convenient for our purposes. Eventually, however, I will replace quantifier raising in section 7 with a continuation-based combinatory categorial grammar for reasons discussed in detail there.

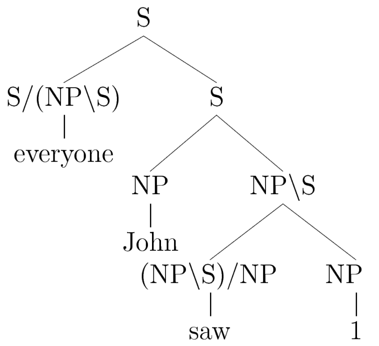
Heim and Kratzer motivate quantifier raising as a strategy to repair type clash. As we saw in (26), a quantificational or lifted NP can occur in subject position without any problem, since a verb phrase is exactly the right sort of object to satisfy the needs of the lifted NP. But when generalized quantifier NPs occur in non-subject positions, type clash can occur.

(29)



That is, *saw* expects a simple NP to its right, but finds instead a generalized quantifier. Since plain LIFT will not help here, we need some other way. Quantifier Raising is the standard technique to resolve this type clash. The basic idea is to raise the generalized quantifier to adjoin to its scope target, replacing the raised quantifier with a variable over simple NP denotations (as Heim and Kratzer have it, in this case the variable will be the numeral 1).

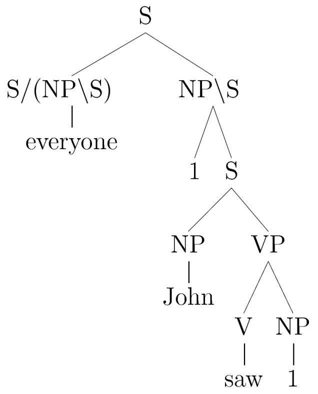
(30)



This resolves the type clash at the level of the transitive verb, since now the verb finds just the type of argument it was hoping for (i.e., NP).

Unfortunately, quantifier raising resolves type clash lower down only to recreate it higher up. At the adjunction site, the generalized quantifier expects an argument of category NP\S, but finds instead an expression of category S. Interestingly, Heim and Kratzer propose a slightly different version of Quantifier Raising that introduces an intermediate node in between the scope target and the result category:[5]

(31)



Following Heim and Kratzer, we can articulate the raising operation into the following steps:

(32) (i) Replace the scope-taking expression with a variable.
(ii) Adjoin the scope-taking expression to its scope target.
(iii) Adjoin a second occurrence of the variable inserted in step (i) to the scope target.

Even these steps do not alleviate all type clashes, since step (iii) creates a subtree with an index (interpreted as a variable) of type `e` as sister to an S node of type `t`: neither one has a type appropriate to serve as a function on the other. Therefore, as Heim and Kratzer explain, the node inserted in step (iii) receives a special interpretation: this constituent translates as a lambda abstract with the index serving

[5] I will sometimes use VP as an abbreviation of NP\S, and V as an abbreviation for (NP\S)/NP to keep the trees a little bit simpler.

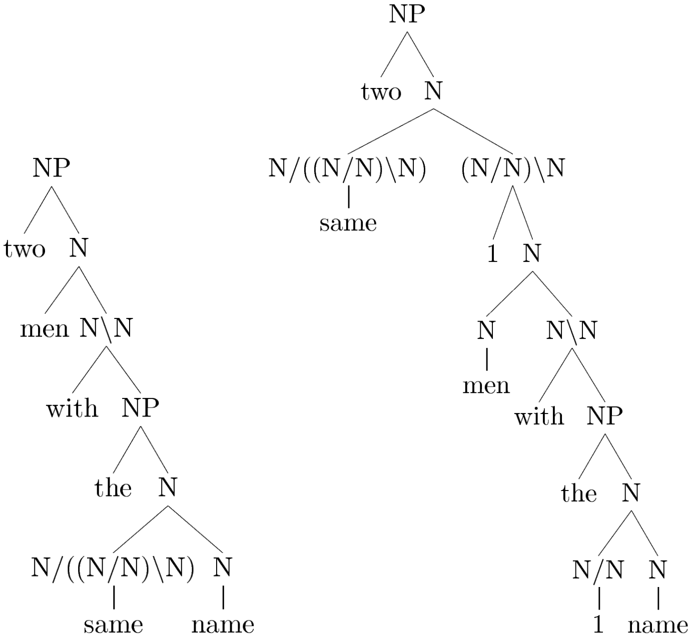
as the distinguished variable. (To visualize the interpretation of the constituent in question, just draw a λ to the left of the uppermost occurrence of the variable.)

In any case, what is most important for present purposes is that the interpretation of the node inserted in step (iii) will have just the right semantic type to serve as an argument of the scope-taking expression. The importance of this extra node is that it will play a crucial role in the discussion of parasitic scope in section 6. Heim and Kratzer do not provide their intermediate node with a category label, so we are free to asign it to category NP\S, since that is the category that the generalized quantifier to its left is looking for.

In general, the raising algorithm will produce analogous results for any scope-taking category q(P,T,R): after inserting a variable, raising, and abstracting, the node inserted in step (iii) will have semantic type $\langle P', T' \rangle$, which is just the right sort of object to serve as the argument to the denotation of the scope-taking element (whose semantic type, recall, is $\langle\langle P', T'\rangle, R'\rangle$).

In particular, since our LIFTed adjective has category q(Adj, N, N) = N/(Adj\N), it functions locally as an adjective, takes scope over a nominal, and returns a nominal as a result:

(33)



On the left, before Quantifier Raising, there is type clash at the level of the adjective. On the right, after Quantifier Raising, the adjective has found its scope.

Thus giving *same* a LIFTed category automatically predicts that it can take scope over a nominal.

5.5. Denotation for same. If we LIFTed and raised a typical intersective adjective such as *long*, the semantics of lifting in combination with the semantics of raising cancel out, in the same way that quantifier raising a LIFTed proper name makes no detectable semantic difference. It is only when the meaning of the NP in question is essentially quantificational (e.g., *everyone*) that quantifier raising makes a detectable difference. Just so, raising an adjective only makes a detectable difference if the adjective's meaning is essentially quantificational.

I am proposing, of course, that *same* is just such an adjective:

(34) a. $\text{type}(\textit{same}) = \text{type}(q(\text{Adj,N,N})) = \langle\langle \textit{Adj}', N'\rangle, N'\rangle$

b. $[\![\textit{same}]\!] = \lambda F_{\langle \text{Adj,N}\rangle} \lambda X_{\texttt{e}}.\exists f_{\texttt{choice}} \forall x < X : Ffx$

Here F is a variable over objects of type $\langle \text{Adj}, \text{N}\rangle$, the type of a function from adjective meanings to nominal meanings, i.e., semantic type $\langle\langle\langle \texttt{e},\texttt{t}\rangle, \langle \texttt{e},\texttt{t}\rangle\rangle, \langle \texttt{e},\texttt{t}\rangle\rangle$. In addition, f is a variable over choice functions of type $\langle\langle \texttt{e},\texttt{t}\rangle, \langle \texttt{e},\texttt{t}\rangle\rangle$. Usually, choice functions are given type $\langle\langle \texttt{e},\texttt{t}\rangle, \texttt{e}\rangle$, but as explained above in section 5.1, we need a function that will deliver a result that is suitable to combine with a determiner; therefore a choice function here will be a nominal modifier that takes (the characteristic function of) a set of individuals and returns a singleton set whose unique member is chosen from the original set. For instance, if *name* denotes a property that is true of the names *Anna*, *Bill*, and *Cam*, then $f([\![\textit{name}]\!])$ might be the property that it true only of the name *Bill.* Finally, I will use capital X as a variable over non-atomic entities. For example, if $X = [\![\textit{Anna and Bill}]\!] = \mathbf{a} \oplus \mathbf{b}$ in the usual way, then $\mathbf{a} < X$ and $\mathbf{b} < X$, where $<$ is the proper-part relation over the count domain.

Inserting the denotation in (34b) into the analysis in (31), we have:

(35)a. $[\![\textit{two men with the same name}]\!] =$

b. $\mathbf{two}(\lambda X.\exists f \forall x < X : [\mathbf{with}(\mathbf{the}(f(\mathbf{name})))\ (\mathbf{men})]\ (x))$

Then a sentence such as *Two men with the same name left* can be paraphrased as follows: There is an object X with cardinality 2 such that there is a choice function f such that each proper subpart of X has $f(\mathbf{name})$, and X left.

This analysis builds the distributivity that Carlson (1987) argues characterizes the internal use of *same* directly into its lexical meaning. Unlike Carlson, however, there is no direct reference to events or sets of events. This is a good thing, of course, in view of NP-internal examples such as (35)!

5.6. Why same *is obliged to take non-trivial scope.* The analysis here says that *same* needs to take scope at some dominating nominal. But I have assumed that an adjective is a nominal modifier, i.e., has category N/N. That means that in an NP such as *two men with the same name*, there are two nodes labelled N that dominate *same*: *men with the same name*, and *same name*, the nominal complement of *the*, as illustrated in (33).

But if the N node corresponding to *same name* were a legitimate scope target for *same*, we would expect **two same men left* to be grammatical with an internal reading.

What, then, prevents *same* from adjoining to its mother? The answer I will offer is that nothing prevents this from the point of view of scope-taking; but because of details of the denotation of *same*, the meaning that results is guaranteed to be true of no object, and hence will be useless. More specifically, the property denoted by *same men* would be true of a non-atomic entity X just in case there is some choice function f and every proper subpart of X is $f(\mathbf{men})$. For the sake of discussion, choose $X = [\![\textit{Bill and Cam}]\!] = \mathbf{b} \oplus \mathbf{c}$. No matter what choice function we select as the value of f, we have $\{\mathbf{b}\} = f(\mathbf{men})$ and $\{\mathbf{c}\} = f(\mathbf{men})$. But since f is a choice function, it follows that $\mathbf{b} = \mathbf{c}$, contrary to the entailment that X is non-atomic, i.e., has distinct atomic parts. By analogous reasoning, there is no suitable choice for X, and the property is semantically guaranteed to denote the empty property in all situations.

In fact, there are other quantificational adjectives that are perfectly happy taking trivial scope.

(36) a. two men with different names
b. two different men

For instance, in addition to the quantificational internal reading of (36a), in which *different* takes scope over the nominal *men with _ names*, in (36b), *different* takes trivial scope over only *men.* (I suspect that this is what Carlson (1987) calls the 'various' reading of *different.*)

It might seem that *two different men* means the same thing as *two men*, and we've swung from a contradictory description with *same* to a semantically redundant modification with *different*; but as noted in Barker (1998), there are situations in which trivial-scope *different* has a non-trivial semantic effect:

(37) a. 4000 ships passed through the lock last year. [Krifka]
b. 4000 different ships passed through the lock last year.

As Krifka (1990) observes, (37a) can be true if 2000 distinct ships passed through the lock twice each, in which case what we have 4000 of are ship stages. In (37b), the addition of *different* renders the stage-based interpretation unavailable: there must be at least 4000 distinct ships. The ships must be different at the level of the individual, and two stages of the same ship do not count as different.

For an example of a quantificational adjective even closer to *same*, consider *similar*, which clearly has a quantificational interpretation as in *Two men with similar names left* (by an argument closely analogous to the discussion of *same* above). But *similar* also has a use with trivial scope, as in *Two similar men left.* The relevant difference between *same* and *similar* is that two distinct atomic entities can fall within the (vague) tolerance of counting as sufficiently similar, without counting as the same.

5.7. A definiteness puzzle. Why does *same* require the definite determiner? That is, why does English insist on **the** *same name* rather than **a** *same name*? Even more peculiar, definite descriptions involving *same* do not trigger existence presuppositions the way that typical definite descriptions do.

(38) a. John and Bill read the long book.
b. John and Bill didn't read the long book.
c. Did John and Bill read the long book?
d. John and Bill might have read the long book.

In (38a), a use of the definite description *the long book* presupposes the existence of a long book. Thus whether the sentence is negated (38b), questioned (38c), or embedded beneath an epistemic modal (38d), the implication remains that a (unique) long book exists.

(39) a. John and Bill read the same book.
b. John and Bill didn't read the same book.
c. Did John and Bill read the same book?
d. John and Bill might have read the same book.

But if the non-quantificational adjective *long* is replaced with *same*, the presupposition disappears: whether or not there is a (unique) book that John and Bill each read is precisely what is at issue in (39a), so if the sentence is negated, questioned, or embedded under an epistemic modal, there is no guarantee that such a book exists.

The quantificational analysis can provide some insight, at least at a functional level. Consider once again the proposed denotation for *two men with the same name*:

(40) $\mathbf{two}(\lambda X.\exists f \forall x < X : [\mathbf{with}(\mathbf{the}(f(\mathbf{name})))\,(\mathbf{men})]\,(x))$

Since f is a choice function, $f(\mathbf{name})$ is semantically guaranteed to denote a property that is true of a unique name. It makes a certain amount of sense that when a nominal is semantically guaranteed to denote a singleton set that the determiner would be *the* rather than *a*; but even if so, it remains a mystery why the presupposed part of the existence implication normally associated with a use of the definite determiner is suspended in the case of *same*.

6. Parasitic Scope

This section shows how the analysis for the NP-internal use easily accounts for more standard examples such as *Everyone read the same book.* The key insight is that, when raising an NP, the extra node inserted by Heim and Kratzer-style quantifier raising has the same semantic type as a nominal, namely, $\langle \mathtt{e}, \mathtt{t} \rangle$. I will exploit this fact by allowing *same* to take the extra node as its scope target.

In order to make this idea work, we need to identify the category N with the category NP\S. But we chose N arbitrarily as a basic syntactic category, so nothing prevents us from choosing N = NP\S.[6]

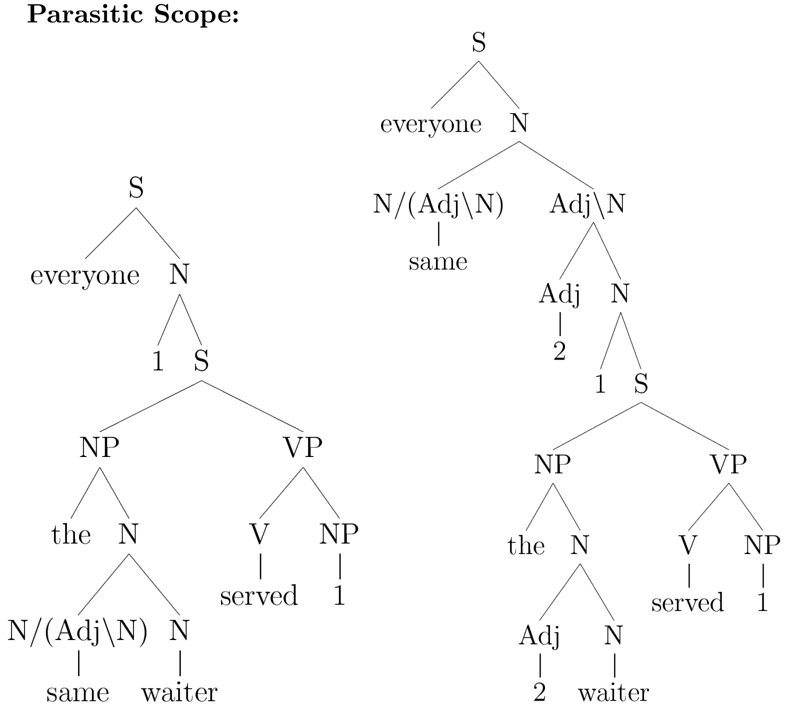
Once we agree that N is an abbreviation for NP\S, parasitic scope falls out:[7]

[6] Ultimately, it will be necessary to distinguish syntactically between nominals (category N),verb phrases (NP\S), and the node inserted during quantifier raising (currently, also NP\S). The analysis in section 7 will make these distinctions; nevertheless, it is not misleading to collapse those distinctions here for the sake of exposition, since the final analysis in section 7 will still arrive a single lexical entry for *same* that simultaneously covers NP-internal examples as well as the examples discussed in this section.

[7] The key example developed in this section will be a type of sentence attributed to Heim in Dowty (1985). Similar examples appear in Stump (1982) without discussion. One point of interest with respect to this sentence is that the trigger (*everyone*) is asymmetrically c-commanded by the NP containing *same*. This is one type of example that makes it more difficult to attempt to reduce the behavior of *same* or *different* to reflexives or anaphors such as *each other* (see Beck (2000) for an instance of this type of analysis for *different*).

(41) The same waiter served everyone. [Stump; Heim]

Parasitic Scope:



In the tree on the left, *everyone* has already taken scope, creating an intermediate node labelled N. This newly-created node then serves as the scope target for *same*, which then raises to yield the tree on the right.

The reason I call this parasitic scope is that the scope target for *same* does not even exist until *everyone* takes its scope. The adjective then hijacks the scope of *everyone*, intervening between the quantifier and what would otherwise be its semantic argument.

Because the semantic argument of the generalized quantifier is $\langle \mathtt{e}, \mathtt{t} \rangle$, the same type as a nominal, we can use the same denotation for *same* proposed above in (34b). More specifically, we have:

(42) a. The same waiter served everyone.

b. $\mathbf{everyone}(\lambda X.\exists f \forall x < X : \mathbf{served}(x)(\mathbf{the}(f(\mathbf{waiter}))))$

c. Everyone collectively has the property of being a group such that there is a unique waiter who served each member of the group.

This analysis assumes that *everyone* denotes a generalized quantifier that can take a property of non-atomic entities as its argument. This somewhat problematic assumption is discussed in some detail in section 6.2; but first, I'll show how the same analysis accounts for plural NP triggers without any further stipulation.

6.1. Plural triggers. Plural NPs are the prototypical triggers for licensing an internal reading for *same* or *different*:

(43) a. The men read the same book.
     b. John and Bill read the same book.

Here is how the assumptions defended so far account for examples like (43): obviously, plural NPs like *John and Bill* and *the men* are NPs. Therefore they can undergo LIFT just like a proper name such as *John.* In some theories, such as Partee and Rooth (1983), they not only can but must undergo LIFT in order to coordinate with generalized quantifiers, as in *every woman and the men.* Unlike Partee and Rooth, but like, e.g., Jacobson (1999), I will assume that LIFT applies freely unless something blocks it.

The net result is that plural NPs are always members of the quantificational category S/(NP\S), and therefore can freely undergo quantifier raising. Normally, quantifier raising an individual-denoting expression has no effect on the final result. But here raising a plural NP does make a detectable difference, since it provides a target for *same* to take scope at.

Thus we automatically predict that, for instance, (43b) has an internal reading that entails that there is a unique book such that proper subparts of the non-atomic entity consisting of John and Bill each read that book, as desired.

Nothing in the reasoning just given was specific to plural NPs. We might expect, then, that the same derivation applies to singular NPs; and in fact, it does:

(44) a. John read the same book.
     b. $\mathbf{John}(\lambda X.\exists f \forall x < X.\mathbf{read}(\mathbf{the}(f(\mathbf{book})))(x))$

Like any NP, *John* can LIFT, undergo quantifier raising, and serve as a scope host to parasitic *same.* But the resulting interpretation is incoherent: it entails that there is a book such that each of the proper subparts of John read that book. Since $<$ gives dominance in the count domain (not the mass domain), John has no suitable proper subparts. The explanation for why (4a) does not have an discernible internal reading, then, is that there are no subparts for the distributivity that is built into the denotation of *same* to quantify over.

6.2. A puzzle for forging a formal link: each. As mentioned above, although the semantics of *same* comport beautifully with fairly standard assumptions about the denotations of plural NPs, they require some less standard assumptions about the denotations of generalized quantifiers like *everyone* and *each person.*

The problem is that because *same* is inherently distributive, it returns a property that is true only of non-atomic entities. But quantifiers like *everyone* are usually assumed to quantify over atomic individuals.

Fortunately for the analysis here, the quantifiers in question for the most part are also compatible with other collective predicates.

(45) a. #John gathered in the living room.
     b. Everyone gathered in the living room.
     c. No one gathered in the living room.

The truth conditions of (45b) require that there is a non-atomic entity X such that every relevant person is a part of X and X has the property of gathering in the living room. But these are exactly the truth conditions we were wishing for for *same*. I will assume (perhaps over-optimistically) that any analysis that will provide the right truth conditions for (45b) will automatically explain the corresponding problematic example involving *same*.

Unfortunately, any explanation for (45b) is unlikely to extend to examples involving *each*.

(46) a. *Each person gathered in the living room.
b. The men (*each) gathered in the living room.

Unlike *everyone*, *each person* does not accept a predicate that is true only of non-atomic entities, whether *each* appears in determiner position as in (46a), or floated as in (46b). Apparently, *each* truly does insist on quantification over atomic entities.

(47)a. Each student follows the same core curriculum.
b. In a cooperative approach to the sponge activity,
you can furnish each student with the same tessellating shape.
c. The students each read the same book.

It is all the more surprising, then, that determiner *each* seems to be compatible with a internal reading with *same* as in (47a) and (47b), though the internal reading seems somewhat degraded with floated *each* as in (47c).

I will sketch a tentative solution here for the problem posed by *each*. Intuitively, it is reasonably clear why (46a) is bad when (47a) is good: *each* in (46a) requires every member of the relevant set of individuals to have the property of gathering in the living room. Since single individuals cannot gather, (46a) is deviant. But *each* in (47a) requires that every relevant individual has the property of reading a certain book, and that is a property that the individuals involved do in fact have.

We can express the fact that the reading property goes all the way down to individuals by adding sensitivity to COVERS (in the sense of Schwarzschild (1996)) to the denotation for *same*. There is independent motivation for doing so:

(48) a. The men and the women gathered in different rooms.
b. The men gathered in different rooms.

(Similar examples can be constructed using *same*, but the truth conditions are clearer with *different*; see section 8 below for a denotation for *different*.) There is a salient reading of (48a) on which the men gathered in one room, and the women gathered in some other room. Certainly it was not the individual men and women who were gathering; instead, the group consisting of all the men and all the women must be divided up somehow into subgroups. Given the conjunction, one salient division puts the men in one subgroup and the women in the other subgroup. Thus in (48), we must assume some way of dividing up the people into (non-atomic) subgroups before we can evaluate whether the sentence is true.

Following Schwarzschild, then, let $\mathbf{Cov}(X)$, a cover over X, be a set of parts of X such that every subpart of X is contained in one of the members of $\mathbf{Cov}(X)$. Then we can consider revising the denotation of *same* as follows:

(49) As in (34): $\lambda F \lambda X \exists f \forall x < X : Ffx$
New: $\lambda F \lambda X \exists f \forall x \in \mathbf{Cov}(X) : Ffx$

The only difference is that instead of quantifying over all the subparts of each group X, we quantify over only certain salient subparts of X, the subparts delivered by the **Cov** function. According to Schwarzschild and others, the **Cov** function is essentially pragmatic in nature, but is nevertheless sensitive to linguistic structure. For instance, the conjunction in (48a) makes the cover consisting of the men and the women highly salient.

All we need assume at this point is that *each*, whether floated or not, somehow forces an atomic cover.

The data in (48) make a compelling case that the distributivity built into scope-taking adjectives must be sensitive to covers, so the refinement in (49) is justified regardless of the treatment of *each.* However, sensitivity to covers does not play an important role in the remainder of the discussion, and I will revert to the original approximation in (48a) for the sake of expository simplicity.

As for *each*, what is left unexplained is how precisely the presence of *each* forces the selection of an atomic cover, though see Beck (2000) for many relevant observations and proposals concerning the selection of covers for examples involving *different.*

7. Continuations

Up to this point I have developed my proposal within a hybrid framework that grafts a Categorial-Grammar theory of categories onto a movement theory of scope-taking. The justification for using CG categories was in order to show how LIFTing explains why it is natural for scope-taking adjectives to target nominals (and indeed, why it is natural for them to take scope in the first place).

The justification for using a movement theory of scope-taking was in order to exploit the familiarity of Quantifier Raising for ease of exposition. Especially intriguing was the discovery that inserting an extra node during Quantifier Raising—what I have been calling "Quantifier Raising Heim and Kratzer style"—provides exactly the right structure to enable parasitic scope.

It would be easy to jump to the conclusion that my analysis of *same* only works in the context of a movement theory of scope-taking. But in fact, the analysis works perfectly well within a system that is DIRECTLY COMPOSITIONAL in the sense of, e.g., Jacobson (1999): each syntactic constituent has a corresponding semantic interpretation, and there is no movement, no quantifier raising, and no LF distinct from surface constituency. (Incidentally, the analysis below is also variable-free, though that fact does not play any significant role in this discussion.)

My movement-free treatment will rely heavily on continuations in order to characterize the behavior of *same* in the most general way. We shall see that in a continuation-based system, there is no need even to stipulate a mechanism equivalent to HK-style node insertion, since I will show that parasitic scope falls out given two independently-motivated assumptions: function composition (the Geach rule) and an Identity rule.

The advantage of building a continuation-based account is that it not only covers NP-internal and standard parasitic uses of *same*, it also accounts for internal readings with non-NP triggers, such as *John hit and killed the same man.*

7.1. Continuations. Continuations are a technique borrowed from computer science for analyzing the semantics of programming languages.

I will not attempt to motivate the theory of continuations here in any great detail; for description, motivation, and history, as well as other linguistic applications of continuations, see Barker (2002), de Groote (2001), Barker (2004), Shan and Barker (2003), and Barker and Shan (2004).

Nevertheless, I will offer a (somewhat limited) analogy from the semantics of focus. In general, the continuation of a subexpression B embedded in a larger expression A is the function mapping objects of type B onto the result of inserting that object in place of B within A. That sounds complicated, but in fact it is a concept familiar from the study of focus, since it is nothing more than the focus denotation of A when B is placed in focus:

$$[\![\textit{John saw } [\textit{everyone}]_F]\!]^{\text{FOC}} = \lambda x.\mathbf{saw}\, x\, \mathbf{j}$$

The focus value of the sentence *John saw everyone* when *everyone* is in focus is the property of being seen by John. This property is also the continuation of the NP *everyone* with respect to the sentence.[8]

The connection between continuations and quantification is immediate: not coincidentally, the semantic argument of the generalized quantifier **everyone** after quantifier raising is this same property, as underlined here:

$$[\![\textit{John saw everyone}]\!] = \mathbf{everyone}(\underline{\lambda x.\mathbf{saw}\, x\, \mathbf{j}})$$

In general, the argument of a scope-taking expression is its continuation with respect to the expression over which it takes scope. In other words, the problem of accounting for in-situ quantification is equivalent to the problem of providing each expression with access to its own continuation. This is why a system that explicitly recognizes and manipulates continuations is ideally suited to reasoning about scope taking.

7.2. Modes of combination. In the discussion above, the category NP\S served as the category of the argument to a generalized quantifier. However, this category also served for labelling verb phrases. It was convenient to pretend that a single category would suffice for both types of object for the sake of simplicity; but NP continuations are not the same thing as verb phrases, and we would have eventually needed to distinguish between them in any framework. Therefore we will provide a new mode of combination specifically for manipulating continuations: in addition to A\B and B/A, we now have A\\B and B//A. (This is a new MODE in the sense of Type-Logical grammar; see Barker and Shan (2004) for an extended discussion of continuations in multi-modal Type-Logical Grammar.) The semantic types of these continuation-slashed categories are the same as their respective plain slashed categories, namely, $\langle A', B' \rangle$.

7.3. Type-shifters for in-situ quantification. We need only add a mechanism for scope-taking to replace Quantifier Raising. This will be accomplished by providing

[8] I am not claiming that there is a direct connection between scope-taking and focus. Rather, both scope-taking and focus deal in continuations; see Barker (2004) for a sketch of what a continuation-based theory of focus might look like.

type-shifting operators. Just as we needed a LIFT for the plain slash, we will need an exactly analogous 𝕃ift for the continuation slash:

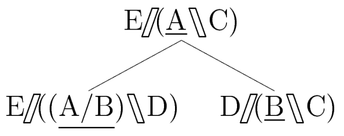
(50)

Lift	$(\mathrm{B}/(\mathrm{A}\backslash\mathrm{B}))\ /\ \mathrm{A}$	$\lambda F\lambda x.Fx$
𝕃ift	$(\mathrm{B}/\!\!/(\mathrm{A}\backslash\!\!\backslash\mathrm{B}))\ /\ \mathrm{A}$	$\lambda F\lambda x.Fx$
𝔻one	$\mathrm{A}\ /\ (\mathrm{A}/\!\!/(\mathrm{B}\backslash\!\!\backslash\mathrm{B}))$	$\lambda F.F(\lambda x.x)$
ℂompose	$((\mathrm{E}/\!\!/(\mathrm{A}\backslash\!\!\backslash\mathrm{C}))\ /\ (\mathrm{D}/\!\!/(\mathrm{B}\backslash\!\!\backslash\mathrm{C})))\ /\ (\mathrm{E}/\!\!/((\mathrm{A}/\mathrm{B})\backslash\!\!\backslash\mathrm{D}))$ $\lambda L\lambda R\lambda\kappa.L(\lambda l.R(\lambda r.\kappa(lr)))$	

In addition to the two versions of lifting, there are two other type-shifters in (50), namely, 𝔻one and ℂompose. What goes up must eventually come down, so after 𝕃ifting builds more complex categories, 𝔻one simplifies them, typically as one of the last steps in a derivation.

The ℂompose rule (whose abbreviation 'ℂ' also stands for 'ℂontinuation') is the main rule governing continuations. The operator in (50) is doubly curried, so it may be more helpful to think of the ℂompose rule as licensing subtrees that match the following form:

(51)



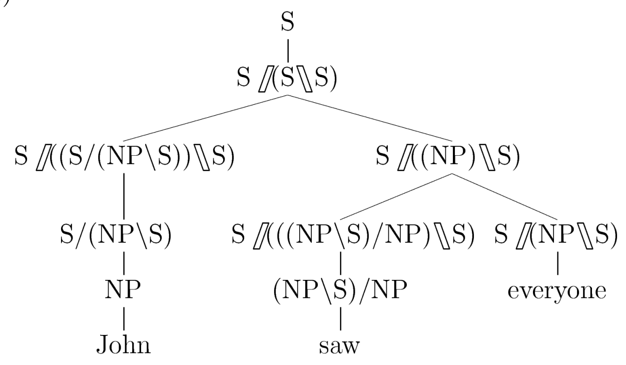
Focusing on the underlined parts, you can discern buried within these categories normal categorial cancellation: A/B followed by B yields A. Thus the ℂompose rule governs syntactic combination in the presence of continuations.

An example will show how a quantificational NP in direct object position takes semantic scope over an entire clause.

john	NP
saw	(NP\S)/NP
everyone	$\mathrm{S}/\!\!/(\mathrm{NP}\backslash\!\!\backslash\mathrm{S})$

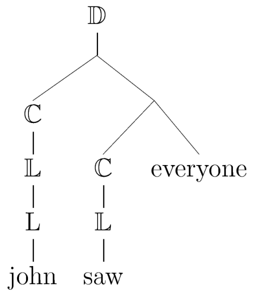
These lexical items give rise to the following derivation:

(52)



In this tree, each branching node is licensed by the ℂompose schema. The nodes with exactly one daughter are licensed by the other type-shifting schemata as follows:

(53)



$$\mathbf{everyone}(\lambda x.\mathbf{saw}\, x\, \mathbf{j})$$

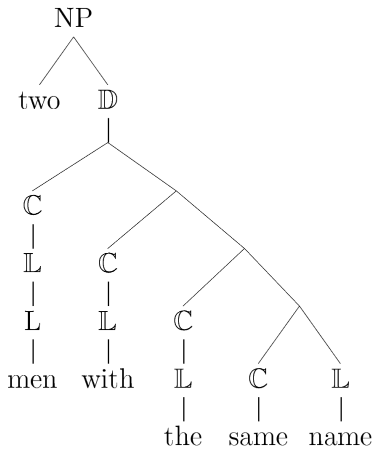
Because *everyone* lives at a higher type (i.e., because it is essentially quantificational), it is necessary to 𝕃ift *saw* before it can combine with *everyone* via the ℂompose rule.

Note that the quantificational expression *everyone* does not undergo movement. Furthermore, the verb phrase *saw everyone* is a constituent in the final derivation, just as in the surface syntax. In general, this system strictly maintains direct compositionality: anything that is a constituent in the syntax is a constituent in the semantics as well, and vice-versa.

7.4. NP-internal same. At this point we can provide an account of NP-internal *same* using almost the same category and denotation proposed above in (34), differing only in the substitution of continuation mode slashes where appropriate:

(54)	two, the	N/NP
	same	N⫽((N/N)⑊N)
	men, name	N
	with	(N\N)/NP

(55)



$$\mathbf{two}(\mathbf{same}(\lambda f \lambda x.[\mathbf{with}(\mathbf{the}(f\, \mathbf{name}))\, (\mathbf{men})](x)))$$

Once again the constituency is the natural syntactic constituency, and *same* takes semantic scope without movement.

7.5. Parasitic scope without movement. Just as above in section 6, the same lexical entry for *same* works for parasitic scope (remembering that N now abbreviates $\mathrm{NP}\backslash\!\!\backslash\mathrm{S}$ rather than $\mathrm{NP}\backslash\mathrm{S}$).

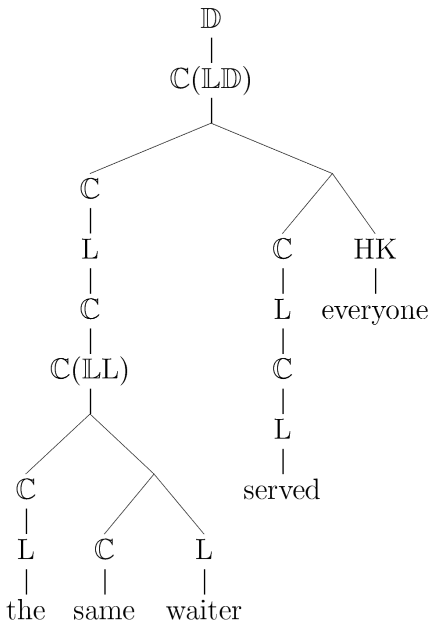
In order to arrive at parasitic scope, we must simulate Heim and Kratzer style quantifier raising, i.e., inserting an extra node in between the scope target and the result category. Ideally, we would need nothing special beyond the basic general mechanism for taking scope. We shall come remarkably close to the ideal here: parasitic scope will follow from two additional assumptions, each of which are motivated by phenomena independent of *same.* But for the initial discussion it will be helpful to consider a type-shifting operator that implements Heim and Kratzer-style QR directly, then explain later how the operator can be factored into simpler, independently-motivated operations.

(56) HK: $(\underline{\mathrm{S}} \mathbin{/\!\!/} ((\mathrm{N} \mathbin{/\!\!/} (\underline{\mathrm{NP}\backslash\!\!\backslash\mathrm{S}})) \mathbin{\backslash\!\!\backslash} \mathrm{N})) \;/\; (\underline{\mathrm{S}/\!\!/(\mathrm{NP}\backslash\!\!\backslash\mathrm{S})})$

Here's how to understand the HK operator (named in honor of Heim and Kratzer, naturally). On the right side of the plain slash ('/') is the category of a generalized quantifier, i.e., $\mathrm{S}/\!\!/(\mathrm{NP}\backslash\!\!\backslash\mathrm{S})$ (the rightmost underlined portion); thus HK takes a plain generalized quantifier as an input. It returns as a shifted result a more complex scope-taking category that splits up the generalized quantifier type into two parts: S on the left (the leftmost underlined portion of (56)), and $\mathrm{NP}\backslash\!\!\backslash\mathrm{S}$ in the middle (the second underlined portion). Sandwiched in between are two occurrences of the category we want to insert (in this case, N). The first (leftmost) occurrence builds a (virtual) node labelled N above the scope target node S; and the second occurrence looks for the newly-constructed N node, above which it constructs the final result S.

It is important not to take the movement metaphor in the previous paragraph too seriously: there is no "movement" and no "building". Rather, as always with a combinatory categorial grammar, there is only application of type-shifting operators.

(57)



$\mathbf{everyone}(\mathbf{same}(\lambda f\lambda x.\mathbf{served}\,x\,(\mathbf{the}(f\,\mathbf{waiter}))))$

In this derivation, HK applies to the trigger *everyone* in order to allow *same* to take parasitic scope.[9]

7.6. Factoring HK. The HK operator in (56) is nothing more than a simple-minded implementation of Heim and Kratzer-style quantifier raising. However, as mentioned above, we can reduce (56) to simpler, independently-motivated factors.

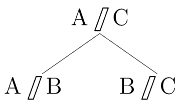
One factor is function composition, for which there is abundant motivation in the literature. For instance, function composition provides an analysis of Right Node Raising such as *John bought and Mary sold the record*, as well as other types of so-called non-constituent coordination (Steedman (1985), Dowty (1988, 1997)). Jacobson (e.g., 1992, 1999) provides a wide variety of additional empirical arguments in favor of adopting function composition, so it is reasonable to assume that any adequate grammar will provide some kind of function composition. The specific instance we will need is:

(58) Geach: $((\mathrm{A}/\!\!/\mathrm{C})\ /\ (\mathrm{B}/\!\!/\mathrm{C}))\ /\ (\mathrm{A}/\!\!/\mathrm{B})$ $\qquad \lambda f\lambda g\lambda x.f(gx)$

Following Jacobson, I will call this curried version of function composition the Geach rule. Once again, it may be easier to understand function composition by considering the sub-trees licensed by this rule:

[9] Unlike previous derivations, here some of the type-shifting operators (e.g., $\mathbb{C}(\mathbb{L}\mathbb{D})$) are complex, formed by applying type-shifters directly to other type-shifters. But because the type shifting operators apply freely, and because each shifter is (deliberately) expressed in the form of a category, nothing prevents this; see (61) for an example.

(59)



The Geach rule expresses the sense in which categorial combination is transitive: if an expression of category A⫽B (something that needs a B to yield an A) is followed by an expression of category B⫽C (something that needs a C to form a B), then these two expressions form a sequence that only needs an C in order to yield a A. In other words, treating the slash as a division symbol (which is where the slash symbol came from in the first place), we have $\frac{A}{B} \times \frac{B}{C} = \frac{A}{C}$. Like LIFT and 𝕃ift, function composition is a theorem in the Lambek calculus and therefore in type-logical grammar (see, e.g., Jäger (2001:42) for derivations and discussion).

In addition to the Geach rule, the only other factor we need is an identity category:

(60) 𝕀dentity: A⫽A $\lambda x.x$

Once again thinking of the slash as a division sign, it is easy to see why this is an identity category, since $\frac{A}{A} \times A = A \times \frac{A}{A} = A$. Allowing the identity type is like allowing multiplication by 1: multiplying by 1 is guaranteed to not change the value of an expression, but nevertheless can be useful for simplifying certain types of algebraic expressions involving fractions.

In Shan and Barker (2003), various instantiations of the identity category are used extensively in the role of an NP gap, in relatively clauses, as WH-trace, etc. In type-logical grammar, identity categories correspond to allowing empty antecedents in logical derivations: hypothesize an A, then immediately discharge the assumption by ⫽-Introduction. Barker and Shan (2004; see especially section 5) discuss linguistic motivation for allowing empty antecedents in Type-Logical Grammar. In other words, there is good reason to allow identity categories as in (60) completely independently of the treatment of *same*.

In any case, if we have Geach and Identity, we can implement the HK operator as follows:

(61)
$$\begin{aligned}
((\mathbb{G}\text{ everyone})\ (\mathbb{L}\mathbb{I})) &= (\mathbb{G}\text{ everyone})\ \big([(\mathrm{D}\sslash(\mathrm{E}\bbslash\mathrm{D}))/\mathrm{E}]\ [\mathrm{A}\sslash\mathrm{A}]\big)\\
&= (\mathbb{G}\text{ everyone})\ [\mathrm{D}\sslash((\mathrm{A}\sslash\mathrm{A})\bbslash\mathrm{D})]\\
&= \big[\big(((\mathrm{A}\sslash\mathrm{C})\ /\ (\mathrm{B}\sslash\mathrm{C}))\ /\ (\mathrm{A}\sslash\mathrm{B})\big)\ (\mathrm{S}\sslash(\mathrm{NP}\bbslash\mathrm{S}))\big]\ (\mathrm{D}\sslash((\mathrm{A}\sslash\mathrm{A})\bbslash\mathrm{D}))\\
&= [((\mathrm{S}\sslash\mathrm{C})\ /\ ((\mathrm{NP}\bbslash\mathrm{S})\sslash\mathrm{C}))]\ (\mathrm{D}\sslash((\mathrm{A}\sslash\mathrm{A})\bbslash\mathrm{D}))\\
&= \mathrm{S}\sslash((\mathrm{A}\sslash\mathrm{A})\bbslash(\mathrm{NP}\bbslash\mathrm{S}))
\end{aligned}$$

In order to get the more specific result corresponding to (56) above, choose A = N = NP⑊S.[10]

In other words, given the scope taking mechanism in (50), along with function composition (Geach) and free insertion of identity categories, parasitic scope falls out for free.

[10] Unlike previous derivations, in this case type-shifting categories combine with a lexical item that precedes it rather than follows it. But once again, nothing prevents this.

7.7. Non-NP triggers. Carlson (1987) emphasizes that *same* and *different* can also distribute over other types of expressions besides NPs:

(62) a. John read and reviewed the same book. V and V
b. John read the same book quickly and thoroughly. Adv and Adv
c. John read the same book every day. Quantificational Adv
d. John usually read the same book. Quantificational Adv

As far as I know, no one has proposed an explicit analysis of *same* when it occurs with non-NP triggers. One advantage of the analysis proposed here is that generalizing to non-NP triggers is quite easy.

In order to generalize to non-NP triggers, it is necessary to generalize the lexical entry for parasitic *same* ever so slightly.

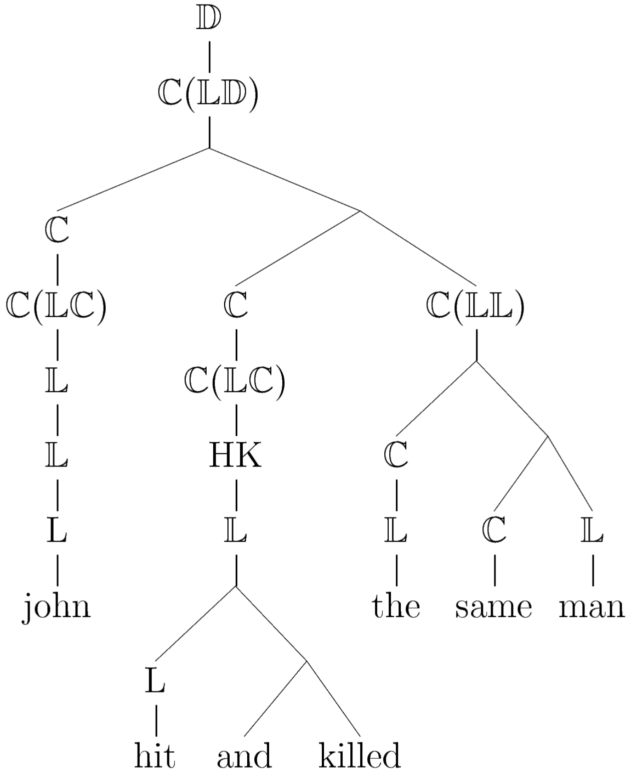
(63) Old: $(\mathrm{NP}\backslash\!\!\backslash\mathrm{S}) /\!\!/ ((\mathrm{N}/\mathrm{N}) \backslash\!\!\backslash (\mathrm{NP}\backslash\!\!\backslash\mathrm{S}))$
New: $(\mathrm{A}\backslash\!\!\backslash\mathrm{S}) /\!\!/ ((\mathrm{N}/\mathrm{N}) \backslash\!\!\backslash (\mathrm{A}\backslash\!\!\backslash\mathrm{S}))$

I have written out out $\mathrm{NP}\backslash\!\!\backslash\mathrm{S}$ rather than the equivalent N for the scope target and the result, since we need to examine the internal structure in order to perceive the full correspondence between the old category and the new one. The only difference is that instead of targeting a category that specifically mentions NP (i.e., the scope target is $\mathrm{NP}\backslash\!\!\backslash\mathrm{S}$ in the original category), the generalized category targets the category $\mathrm{A}\backslash\!\!\backslash\mathrm{S}$, where A is a meta-variable over categories.[11]

In continuation terms, $\mathrm{NP}\backslash\!\!\backslash\mathrm{S}$ is a clause that still needs an NP in order to be complete. Analogously, $\mathrm{A}\backslash\!\!\backslash\mathrm{S}$ is a clause that still needs something of category A in order to be complete, whatever A turns out to be. If we choose A = NP, we generate the standard examples in which the trigger is a plural or quantificational NP. But if we choose $\mathrm{A} = (\mathrm{NP}\backslash\mathrm{S})/\mathrm{NP}$—i.e., the category of a transitive verb—then we generate an instantiation of the schema that is appropriate when the trigger is a coordinate transitive verb:

[11] In a more comprehensive grammar, it would be necessary to distinguish between the category of a nominal and the category $\mathrm{NP}\backslash\!\!\backslash\mathrm{S}$. The generality of this proposed lexical entry for *same* can be preserved, however, as long as the category for nominals is a continuation whose result is S. For instance, we could choose $\mathrm{N} = \mathrm{E}\backslash\!\!\backslash\mathrm{S}$, where E is an abstract category with semantic type $\mathtt{e}$.

(64)



$$((\mathbf{same}(\lambda f(\lambda R.R(\mathbf{the}(f\mathbf{man}))(\mathbf{j}))))(\mathbf{hit} \oplus \mathbf{killed}))$$

Crucially in (64), HK (or, equivalently, its decomposition in terms of $\mathbb{G}$, $\mathbb{L}$, and $\mathbb{I}$) applies to the coordinate transitive verb. This only makes sense if we think of the coordinate transitive verb as potentially taking scope over the clause that contains it. But there is nothing privileged about NPs with respect to the $\mathbb{L}$ift operator. Other expression types can freely undergo $\mathbb{L}$ift and take scope, including transitive verbs. Usually, $\mathbb{L}$ifting a non-quantificational expression has no net semantic effect; but in the presence of *same*, $\mathbb{L}$ifting the coordinate verb does have a detectable semantic effect, since it provides a scope target for *same* to take parasitic scope.

Thus the analysis here makes concrete and explicit the exact analogy between the way in which conjoined NPs give rise to an internal reading with *same* and the way in which conjoined transitive verbs (and other triggers) do.

7.8. Generalized distributivity. Generalizing the semantics is trivial.

(65) $[\![\mathit{same}]\!] = \lambda F_{\langle \mathrm{Adj,N} \rangle} \lambda \mathcal{X}.\exists f_{\texttt{choice}} \forall x < \mathcal{X} : Ffx$

The only change I have made from the original denotation proposed for NP-internal uses in (34) above is that I have refrained from typing the variable $\mathcal{X}$ (in recognition of which I have set it in a curly face), which we now allow to range over any type. (In computer science jargon, the denotation for *same* is now type-polymorphic.)

In order for this to work, we must assume that the relevant semantic domains have a boolean structure (in the sense of Keenan and Faltz (1985)), including at least closure under a join operation—i.e., that there is a transitive verb denotation $[\![\textit{read}]\!] \oplus [\![\textit{reviewed}]\!]$ which is the join of the transitive verb meanings $[\![\textit{read}]\!]$ and $[\![\textit{reviewed}]\!]$.

In addition, the part relation $<$ must be well-defined over each of the domains that are capable of triggering an internal reading for *same*. In the example at hand, we must assume that $<$ is the dominates relation induced by the boolean join structure over the set of transitive verb meanings, so that if *hit and killed* denotes the complex relation $[\![\textit{hit}]\!] \oplus [\![\textit{killed}]\!]$, then $[\![\textit{hit}]\!] < ([\![\textit{hit}]\!] \oplus [\![\textit{killed}]\!])$ and $[\![\textit{killed}]\!] < ([\![\textit{hit}]\!] \oplus [\![\textit{killed}]\!])$. This predicts the following truth conditions:

(66) a. John hit and killed the same man.

b. $((\mathbf{same}(\lambda f(\lambda R.R(\mathbf{the}(f\mathbf{man}))(\mathbf{j}))))(\mathbf{hit} \oplus \mathbf{killed}))$

c. There is a non-atomic relation of (hitting ⊕ killing) such that
there is a choice function f such that
for all proper parts α of (hitting ⊕ killing),
John did α to f(man).

As noted by Carlson (1987), the distributive quantification built into the meaning of *same* only quantifies over verbal meanings that correspond to distinct events. That is, although it is possible to describe a single striking event as both a hitting and a killing (*David hit and killed Goliath*), the internal reading of *same* forces the subparts of the complex relation to correspond to separate events. Thus on the internal reading of (66), the hitting and the killing must be two distinct events. But this is what we would expect if the quantification built into the meaning of *same* behaves like normal quantification, which always quantifies over distinct elements in any domain.

7.9. Short digression on buying versus selling. The way that the distributivity built into *same* insists on distinct events bears on some long-standing issues relating to the nature of events. On a neo-Davidsonian conception of thematic relations (e.g., Parsons 1990), each event must have a unique agent, a unique theme, and so on. Since every buying event is necessarily also a selling event, and since the agent of a buying event is the recipient of a selling event, it follows that buying and selling must count as distinct events, despite necessarily occupying the same physical space during the same moments of time.

(67) John bought and Mary sold the same book.

This sentence has an internal interpretation on which John was in the habit of buying a certain type of book and Mary was in the habit of selling the same type of book. That reading is facilitated (and may even require) that John bought the relevant books from someone other than Mary, and Mary sold her books to someone other than John.

But if buying and selling truly are distinct events, we should expect a different internal reading on which (67) describes a single transaction. Imagine, then, that John bought a book from Mary, and that was the only book John ever bought in his life, as well as the only book that Mary ever sold. Native speakers uniformly report that (67) cannot be used to describe that transaction. The obvious conclusion is that buying and selling do not in fact count as distinct events, at least, not according to the standards of *same*.

8. Conclusions

I have concentrated mostly on *same* in this paper, but Keenan (1992) identifies a number of other constructions that also lie beyond the Frege boundary, some of which may also have compositional semantic analyses along the lines suggested here for *same*. Prime among these, of course, is *different*.

(68) a. John read and reviewed different books.
b. $[\![\textit{different}]\!] = \lambda F.\lambda X \forall f, g \in \texttt{choice}; \forall x, y < X : (Ffx \wedge Fgy \wedge f = g) \rightarrow x = y$

As near as I can tell, this denotation gives reasonable truth conditions for NP-internal uses, parasitic uses distributing over an NP denotation, as well as uses with a non-NP trigger as in (68a).

Keenan also draws attention to resumptive uses of *same* and *different*:

(69) a. The same people ordered the same dishes.
b. Different students answered different questions.

What is intriguing about the internal reading of (6a) is that not only must the same group of people end up eating the same collection of dishes, each person in the group must eat the same dish that that person ate the last time. I will not attempt a complete analysis of resumptive uses here, except to speculate that the first *same* may be a deictic use. In contrast, for (6b), the first occurrence of *different* seems to me to have trivial scope (that is, having scope over the minimal N as discussed above in section 5.6).

Stump, Carlson, Keenan and others observe that *same* and *different* are far from the only adjectives that pose similar challenges for compositional treatments. Other adjectives mentioned by Carlson include *distinct*, *separate*, and *similar*. (Indeed, I suspect that *similar* is a better candidate to serve as the dual of *different* than *same* is.) In addition, other adjectives that may take nominal scope include *identical*, *unrelated*, *mutually incompatible*, and *opposite*.

8.1. Can this analysis be re-translated back into a movement framework? In section 7, I argued that continuations provide an insightful way to express an analysis covering parasitic uses of *same*, both with NP triggers and non-NP triggers. This shows that we can have a scope-taking analysis of *same* in a movement-free, directly compositional system.

What would we have to do to reconstruct the complete analysis within a QR-based movement framework?

The crucial insight relating to continuations is that the semantic scope of an expression is nothing more or less than its continuation with respect to the larger expression over which it takes scope. That means that in order for *same* to take scope that is parasitic on, say, the scope of a conjoined transitive verb, the coordinate verb must somehow take scope over its clause. In the system here, that happens quite naturally: the coordinated verbs undergo $\mathbb{L}$ift, and the combinatorics does the rest.

Thus reconstructing this approach in a movement framework would require two main innovations. First, we would need to introduce a Categorial-Grammar style theory of category labels. In effect, this was done in an informal way in the earlier sections, in which I offered LF trees that contained slashed categories. Note that in addition to just writing down categories that happen to contain slashes, it is necessary to make QR sensitive to details of the categories, so that QR would apply to any category of the shape $\mathrm{R}/\!\!/(\mathrm{P}\backslash\!\!\backslash\mathrm{T})$.

The second innovation is that type-shifting operators including at least $\mathbb{L}$ift would have to be freely available. In section 5, I appealed to Lift in order to motivate a scope-taking lexical category for *same* as natural; however, nothing in that section hinged on assuming that Lift was a productive part of any specific derivation. Nevertheless, it is fairly common to have a movement-based theory that also makes use of type-shifting operations, though usually type-shifting must be forced (rather than optional, as here).

Finally, of course, we would have to stipulate Heim and Kratzer-style Quantifier Raising.

All of this is possible, of course. However, once we have free application of type-shifting operations that produce arbitrarily complex CG-style category labels, adding movement to the theory is unnecessary, since we can instead simply add two type-shifters to handle scope-taking, as shown in section 7.3. In addition to theoretical parsimony (no movement, no level of LF distinct from surface structure), the CCG account enjoys the benefits of direct compositionality, in which each syntactic constituent has a semantic denotation.

8.2. Summary. Whether the analysis makes use of movement and LF or not, the approach here depends on constructing continuations. In particular, whenever *same* takes parasitic scope, its scope must be defined in terms of the continuation of some other expression (the trigger) elsewhere in the clause.

To summarize, I am proposing the following general analysis for *same*:

(70) a. $(\mathrm{A}\backslash\!\!\backslash\mathrm{S}) \;/\!\!/\; ((\mathrm{N}/\mathrm{N}) \;\backslash\!\!\backslash\; (\mathrm{A}\backslash\!\!\backslash\mathrm{S}))$
b. $\lambda F_{\langle \mathrm{Adj},\mathrm{N}\rangle}\lambda\mathcal{X}.\exists f_{\texttt{choice}}\forall x < \mathcal{X} : Ffx$

Given a suitably general theory of scope-taking, and assuming free application of the independently-motivated operators $\mathbb{L}$ift, $\mathbb{G}$each, and $\mathbb{I}$dentity, this single schematic lexical entry accounts for NP-internal uses, parasitic uses with NP triggers, and parasitic uses with non-NP triggers.

Thus continuations provide novel and potentially insightful compositional analyses even in the realm beyond the Frege boundary.

References

Barker, C. 1998. Individuation and Quantification. *Linguistics and Philosophy* **30.4**:683–691.

Barker, C. 2002. Continuations and the nature of quantification. *NALS*.

Barker, C. 2004. Continuations and natural language. in Thielecke, H. (ed), *Proceedings of CW'04*.

Barker, C. and Chung-chieh Shan. 2004. Grafting trees: continuations in Type Logical Grammar. ms.

Barwise, Jon and Robin Cooper: 1981, Generalized Quantifiers in Natural Language, *Linguistics and Philosophy* **4**, 159–200.

Beck, S. 2000. The Semantics of *Different*: Comparison Operator and Relational Adjective. *L&P*.

Carlson, Greg. 1987. *Same* and *Different*: some consequences for syntax and semantics. *L&P*.

Dekker, Paul. 2003. Meanwhile, within the Frege Boundary. *L&P*.

Dowty, David. 1985. A unified indexical analysis of *same* and *different*: a response to Stump and Carlson. ms.

Dowty, David. 1988. Type raising, functional composition, and non-constituent coordination. In Emmon Bach, Richard Oehrle, and Deirdre Wheeler, eds., *Categorial Grammar and Natural Language structures*, 153–198. Dordrecht: Kluwer.

Dowty, David. 1997. Non-Constituent Coordination, Wrapping,and Multimodal Categorial Grammars. In Maria Luisa Dalla Chiara, Kees Doets, Daniele Mundici, and Johan van Benthem, eds., *Structures and Norms in Science, Volume Two of the Tenth International Congress of Logic, Methodology and Philosophy of Science, Florence, August 1995*, 347–368. Dordrecht: Kluwer.

van Eijck, Jan. 2003. Relations, Types and Scoping. Slides.

van Eijck, Jan. 2004. Normal Forms for Characteristic Functions on n-ary Relations. Ms.

de Groote, Philippe. 2001, Continuations, type raising, and classical logic, in R. van Rooy and M. Stokhof (eds.), *Thirteenth Amsterdam Colloquium*, pp. 97–101. Institute for Logic, Language and Computation, Universiteit van Amsterdam.

Heim, I. 1985. Notes on Comparatives and Related Matters. ms.

Heim, I. and A. Kratzer: 1998, *Semantics in Generative Grammar*, Blackwell, Oxford.

Hendriks, H.: 1993, *Studied Flexibility*, ILLC Dissertation Series, Amsterdam.

Jacobson, Pauline. 1992. Antecedent Contained Deletion in a Variable-Free Semantics. In Chris Barker and David Dowty, eds., *Proceedings of SALT II*, Columbus: Ohio State University.

Jacobson, P.: 1999, Towards a variable free semantics. *Linguistics and Philosophy* **22**, 117–184.

Jäger, Gerhard, 2001. *Anaphora and Type Logical Grammar*. Habilitationsschrift.

Keenan, E. 1992. Beyond the Frege Boundary. *L&P*. (Revised version on the web.)

Keenan, Edward and Leonard Faltz. 1985. *Boolean semantics for natural language*. Dordrecht: D. Reidel.

Krifka, Manfred. 1990. Four thousand ships passed through the lock: object-induced measure functions on events. *Linguistics and Philosophy* **13**: 487-520.

Lambek, Joachim, 1958. The mathematics of sentence structure. *American Mathematical Monthly*, **65**: 154–170.

Lasersohn, P. 2000. Same, Models and Representation. SALT 10.

Moltmann, F. 1992. Reciprocals and *Same/Different*: Towards a Semantic Analysis. *L&P*.

Moortgat, Michael. 1997. Categorial type logics. In Johan van Benthem and Alice ter Meulen (eds), *Handbook of Logic and Language*, 93–177.

Nunberg, Geoffrey. 1984. Individuation in Context. *WCCFL* **2**: 203–217.

Parson, Terence. 1990. *Events in the Semantics of English. A Study in Subatomic Semantics*. Cambridge: MIT Press.

Partee, B.H. and M. Rooth. 1983. Generalized conjunction and type ambiguity, in R. Bäuerle, C. Schwarze, and A. von Stechow (eds.), *Meaning, Use, and Interpretation of Language*, pp. 361–383. Walter de Gruyter, Berlin.

Partee, Barbara Hall. 1987. Noun Phrase Interpretation and Type-Shifting Principles. In Groenendijk, J., D. de Jongh, and M. Stokhof, eds., *Studies in Discourse Representation Theory and the Theory of Generalized Quantifiers*, 115–143. Foris.

Schwarzschild, Roger. 1996. *Pluralities*. Dordrecht, Kluwer.

Shan, C., and Barker, C. 2003. Explaining crossover and superiority as left-to-right evaluation. Ms.

Steedman, Mark. 1985. Dependency and coordination in the grammar of Dutch and English. *Language* **61.3**:523–568.

Steedman, Mark. 2000, *The Syntactic Process*, MIT Press, Cambridge, MA.

Stump, G. 1982. A GPSG Fragment for 'Dependent Nominals'. ms.

Tovena, L. & Van Peteghem, M. (in press). Facets of 'different' in French: *différent* and *autre*. Beyssade C. et al (eds.) *Actes de CSSP 2002: Empirical issues in formal syntax and semantics*, Presses Universitaires de la Sorbonne, Paris.