Continuations and the nature of quantification[1]

Chris Barker <barker@ucsd.edu>
Linguistics, University of California, San Diego
http://ling.ucsd.edu/~barker

Abstract: This paper proposes that the meanings of some natural language expressions are best treated as functions on their own continuations. Continuations are a well-established technique in the theory of programming language semantics; in brief, a continuation is the entire default future of a computation. I show how continuations 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 other similar mechanisms used in many recent HPSG, Categorial, or Type-logical treatments), and no need for type-shifting (as in Hendriks' Flexible Types account). On the empirical side, unlike any theory of quantification I know of, the continuation analysis obeys the Syntactic Constituent Integrity Constraint: a quantificational element external to a given syntactic constituent must take scope over all or none of the quantificational elements within that constituent. In particular, because the continuation analysis respects syntactic integrity, it automatically makes accurate predictions related to quantificational elements embedded within NPs in situations that require multiple stipulations in other theories. Continuations also provide a natural account of generalized conjunction that is significantly simpler than other accounts, since the continuation approach does not require type shifting. Compositionality issues are discussed in some detail.

[1] 2 November, 2000: refereeing version plus corrections and minor expository revisions. Thanks to Mark Gawron for comments and advice.

1. Introduction: the Continuation Hypothesis

I will motivate the use of continuations as a tool for describing natural language meaning by showing how a continuation-based analysis can unify several aspects of nominal quantification in a new way. On a descriptive level, we can identify a number of questions whose answers might logically have been independent of one another:

(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?

It may seem strange to discuss scope displacement and scope ambiguity separately, since they cannot be fully independent; after all, scope ambiguity cannot exist without scope displacement, though not vice-versa. My reason for doing so is at least partly expository, though 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 1991:116) 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 LF 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 version for the standard QR view.[1] On their version of the story, non-quantificational NPs denote entities (type e), quantificational NPs (henceforth, 'QNPs') denote generalized quantifiers (type <<e,t>,t>), and typical transitive verbs denote relations over entities (type <e,<e,t>>).[2] When QNPs occur in subject position, type-driven composition allows (indeed, requires) them to take the verb phrase (type <e,t>) as an argument. However, when QNPs occur in non-subject positions, 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,

[1]Heim and Kratzer 1998 is a textbook, and consequently their 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]I will rely throughout this paper on Montague's 1970 notation for semantic types. 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.

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 the QNPs 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. Why are subjects special? 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.

Choosing the other horn of the dilemma, Montague 1970 (henceforth, PTQ) gives a resounding answer to the duality question: non-quantificational NPs and QNPs uniformly 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 any type mismatch. But this renders the existence of a mechanism for scope displacement and scope ambiguity completely mysterious, since it would be much simpler for QNPs to always have fixed scope as determined by the lexical properties of their functor predicate. Once again we fail to provide a unified answer to the three questions in (1).

In Hendriks' 1988, 1993 Flexible Types approach (compare to Keenan 1987 for an interestingly similar idea), many transitive verbs have as their basic type a relation over individuals. He 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 generalized quantifier 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 function as other types of NP, so there is no type clash whatsoever 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 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 an unavoidable consequence of indeterminacy in the way in which continuations get put together in the course of composing a complex meaning.

1.1. Compositionality and Logical Form

QR involves radically rearranging syntactic constituency in order to make semantic interpretation more convenient. Whatever theory-internal or empirical justification there may be for the QR strategy, it requires a drastic compromising of the Principle of Compositionality. QR theories can be compositional (indeed, PTQ itself is a type of QR theory). However, allowing Logical Form to differ significantly from surface syntax seriously weakens the constraints the principle of compositionality places on theories of meaning. (This point is developed more precisely in section 6.) Therefore this paper is in part in service of the Null-LF hypothesis, which is that semantic interpretation proceeds directly from overt syntactic structure.

There are plenty of accounts of quantification that do without LF, but that use some kind of storage mechanism. Such systems are perfectly practical, but also fail to be compositional inasmuch as the interpretation at the level where a stored quantifier is discharged depends on content of a deeply embedded constituent, rather than only on the values of the immediate subconstituents. Cooper storage is the standard example, but typical recent treatments of quantification in HPSG, LFG, CG, and type-logical grammar also fall in this category.

Continuation semantics is compositional in the sense that the meaning of a syntactic constituent is a function only of the meanings of the immediate subconstituents and the way in which they are put together. The Flexible Types approach is the only other theory of quantification I know of that is compositional and consistent with the Null LF hypothesis. Continuations versus Flexible Types differ on other important issues, however, as discussed below, including empirical predictions (see especially section 3).

Before we begin, a note on specific indefinites. There is a long tradition of treating specific indefinites as a species of scope displacement. However, work of Farkas, Fodor and Sag, Abush, Reinhart, Kratzer, Winter, and others converge on the conclusion that the so-called de dicto/de re ambiguity should not be lumped together with quantifier scope. For the purposes of this paper I will assume along with Kratzer 1998:167 that “indefinite NPs are ambiguous between a specific and a quantificational interpretation. If they are quantificational, their scope is local, and they are interpreted as generalized quantifiers, like any other quantifier phrase.” If specific, they are treated differently—in Kratzer’s case, by means of a context-controlled choice function. I develop an explicit treatment of wide-scope indefinites using higher-order continuations elsewhere (Barker ms.).

2. Continuations

Reynolds 1993 gives a detailed history relating how the concept of continuations emerged independently in the work of several computer scientists in the 1960s and early 1970s. According to Danvy and Talcott 1998:115, continuations currently are “ubiquitous in many different areas of computer science, including logic, constructive mathematics, programming languages, and programming.” For instance, continuations have long played a prominent role in the programming language Scheme. The most recent revision of the Scheme language specification (Kelsey et al. 1998:71) explains continuations as follows:

(3) Whenever a Scheme expression is evaluated, there is a *continuation* wanting the result of the expression. The continuation represents the entire (default) future for the computation. If the expression is evaluated at top level, for example, then the continuation might take the result, print it on the screen, prompt for the next input, evaluate it, and so on forever. Most of the time the continuation includes actions specified by user code, as in a continuation that will take the result, multiply it by the value stored in a local variable, add seven, and give the answer to the top level continuation to be printed. Normally these ubiquitous continuations are hidden behind the scenes and programmers do not think much about them. On rare occasions, however, a programmer may need to deal with continuations explicitly.

Continuations, then, in brief, are the entire default future of a computation.

To elaborate on Kelsey et al.'s example, computing the expression "$(7 + ((3 + 1) * x))$" involves computing the subvalue "$(3 + 1)$", determining the value of the local variable x in the relevant context, multiplying the two subvalues together, and then adding the intermediate result to 7. The continuation for the subexpression "$(3 + 1)$" is the fate of the value it returns: that value is going to be multiplied by the value of x, and the result of that computation will be added to 7. In symbols, the continuation of the subexpression "$(3 + 1)$" is $\lambda y.7 + (y * x)$. Each subexpression has its own continuation, including constants: the continuation for the subexpression "7" is $\lambda y.y + ((3 + 1) * x)$.

Kelsey et al. note that virtually any known control structure can be reconstructed using continuations (this is 'control' in the computational sense of control over flow-of-execution). Some examples of control constructions from familiar programming languages include `if-then-else` statements, `goto` statements, `for` loops, `while` loops, the `throw/catch` construction, `return` statements, etc. What is interesting about Scheme is that in addition to normal control mechanisms, it provides a general, explicit way for a function to manipulate its own continuation.

Part of the continuation hypothesis, then, is the claim that quantificational NPs are natural language analogs of control structures. If continuations can unify control structures in programming languages, perhaps they can unify quantification in natural language as well.

Two disclaimers are in order. First, dealing in continuations does not require us to conceive of semantic evaluation as an algorithm whose execution unfolds through time. In fact, Scheme itself has a declarative (i.e., non-procedural) model-theoretic semantics, including an implementation of continuations (given in Kelsey et al. 1998). Second, although I am taking the computational notion of a continuation for my inspiration, I have not tried to remain faithful to the traditional conception. There are a number of important differences between continuations as used here and as used in the description of programming language semantics; to mention just one, I will be treating languages involving ambiguity.

Finally, it is important to bear in mind that continuations per se exist independently of any framework or specific analysis. All occurrences of expressions have continuations in any language that has a semantics, whether that language is artificial or natural. Continuations are just a different way of looking at the relationship between the meaning of a complex expression and the meaning of its parts. 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, but precisely how natural language expressions do or don't interact with them.

2.1. Deriving generalized quantifiers

Consider the following simple, context-free, extensional grammar without quantification:

(4)

	Syntax	Semantics
a.	S $\rightarrow$ NP VP	VP(NP)
b.	VP $\rightarrow$ Vt NP	Vt(NP)
c.	NP $\rightarrow$ John	**j**
d.	NP $\rightarrow$ Mary	**m**
e.	VP $\rightarrow$ left	$\lambda x.\mathbf{left}(x)$
f.	Vt $\rightarrow$ saw	$\lambda x \lambda y.\mathbf{saw}(y, x)$

Leaving the details of the model-theoretic interpretation of the logical language used to express the semantic part of these rules implicit (but, I trust, obvious), we have the following translations after lambda conversion:

(5)

John left.	$\mathbf{left}(\mathbf{j})$
John saw Mary.	$\mathbf{saw}(\mathbf{j}, \mathbf{m})$

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

(6)

Syntactic category	Semantic type	Gloss on the semantic type
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.

(7)

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** will have the semantic type of their corresponding syntactic category (for these examples, e, <e,t>, and <e,<e,t>>, respectively).

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 new lambda-abstracts.) Even so, the implementation here will be complicated enough. Continuations are often presented using an untyped lambda calculus; however, I will carefully type all of the expressions in this initial exposition.

Adding continuations to the basic grammar in (4) requires two new elements that must be carefully distinguished: continuations, and continuized denotations. A continuation, as discussed above, is a function from a normal (uncontinuized) value to the result of the entire computation. For instance, in the uncontinuized grammar in (4), a verb phrase denotes a set of entities, so that type(VP) = <e,t>. Since we will only be evaluating declarative sentences in this paper, the result of the entire computation is always a truth value, so that the type of a VP continuation, which I will write c_{VP}, is <<e,t>,t>. Therefore I will augment the variable symbols in the logical translation language as follows:

(8)

LOGICAL SYMBOL	TYPE	DESCRIPTION
c_{S}	<t,t>	Sentence continuation
c_{VP}	<<e,t>,t>	VP continuation
c_{NP}	<e,t>	NP continuation
c_{Vt}	<<e,<e,t>>,t>	Transitive-verb continuation

A continuized denotation, in contrast, is a denotation that manipulates continuations. I will indicate a continuized denotation by underlining, so that $\underline{\mathrm{VP}}$ is a continuized VP denotation. Continuized denotations are always functions that take a single argument, and that argument will always be a continuation. For instance, the continuized verb phrase meaning $\underline{\mathrm{VP}}$ is a function on VP continuations. The value returned by a continuized denotation is the same kind of value returned by a continuation, in our case, a truth value. Thus a continuized VP is a function from VP continuations to truth values: type($\underline{\mathrm{VP}}$)=<<<e,t>,t>,t>.

(9)

LOGICAL SYMBOL	TYPE	DESCRIPTION
$\underline{\mathrm{S}}$	<<t,t>,t>	Continuized S denotation
$\underline{\mathrm{VP}}$	<<<e,t>,t>,t>	Continuized VP denotation
$\underline{\mathrm{NP}}$	<<e,t>,t>	Continuized NP denotation
$\underline{\mathrm{Vt}}$	<<<e,<e,t>>,t>,t>	Continuized transitive verb denotation

Thus an expression such as "$\underline{\mathrm{VP}}(c_{\mathrm{VP}})$" is well-typed, and denotes a truth value.

With these preliminaries in place, here is one way to continuize the basic grammar in (4):

(10)

	Syntax	Semantics incorporating continuations
a.	S $\rightarrow$ NP VP	$\lambda c_S.\underline{VP}(\lambda P.\underline{NP}(\lambda x.c_S(P(x))))$
b.	VP $\rightarrow$ Vt NP	$\lambda c_{VP}.\underline{NP}(\lambda x.\underline{Vt}(\lambda R.c_{VP}(R(x))))$
c.	NP $\rightarrow$ John	$\lambda c_{NP}.c_{NP}(\mathbf{j})$
d.	NP $\rightarrow$ Mary	$\lambda c_{NP}.c_{NP}(\mathbf{m})$
e.	VP $\rightarrow$ left	$\lambda c_{Vi}.c_{Vi}(\lambda x.\mathbf{left}(x))$
f.	Vt $\rightarrow$ saw	$\lambda c_{Vt}.c_{Vt}(\lambda x \lambda y.\mathbf{saw}(y, x))$

It should be clear how the type of the continuation variables (c_X) follows from the type of other expressions in the rule. Therefore in the remainder of the paper I will often omit the subscript from the continuation variable for the sake of clarity.

The general pattern relating each rule of the basic grammar in (4) to its continuized version in (10) is given as a schema in section 5. In the meantime, 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 (4).

(11)

[S [NP [John]] [VP [left]]]	Syntax for *John left*.
$\lambda c.\underline{VP}(\lambda P.\underline{NP}(\lambda x.c(P(x))))$	Rule (10a).
$\lambda c.\underline{VP}(\lambda P.((\lambda c.c(\mathbf{j}))(\lambda x.c(P(x)))))$	Rule (10c).
$\lambda c.\underline{VP}(\lambda P.c(P(\mathbf{j})))$	λ-conversion
$\lambda c.((\lambda c.c(\lambda x.\mathbf{left}(x)))(\lambda P.c(P(\mathbf{j}))))$	Rule (10e).
$\lambda c.c(\mathbf{left}(\mathbf{j}))$	λ-conversion

According to the continuation grammar, *John left* denotes a function from sentence continuations to a truth value. If we provide ⟦*John left*⟧ with the most trivial continuation possible (the identity function, $\lambda p.p$), we get

(12)

$(\lambda c.c(\mathbf{left}(\mathbf{j})))(\lambda p.p)$	trivial continuation
$\mathbf{left}(\mathbf{j})$	λ-conversion

It is easy to verify for this simple grammar that when sentence denotations are given a null continuation, the continuation grammar computes the same values as the original basic grammar. This equivalence is proven for the general case in section 5.

Note in (9) that a continuized NP denotation is of type <<e,t>,t>. This, of course, is exactly the (extensional version of the) generalized quantifier perspective on entity-denoting NPs like *John* as proposed in PTQ or Barwise and Cooper 1981.

This result bears emphasizing: merely by making continuations explicit, we have derived the generalized quantifier conception of NP meaning. Unlike the use of generalized quantifiers in say, PTQ, nothing in the notion of a continuation is NP-specific. In other words, the conception of NPs as generalized quantifiers is a special case of the more general concept of a continuation. According to the Continuation Hypothesis, then, Montague correctly recognized that QNPs need access to their continuations, but failed to treat continuations systematically throughout the grammar.

2.2. 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 $\rightarrow$ everyone $\lambda c.\forall x : c(x)$
b. NP $\rightarrow$ someone $\lambda c.\exists x : c(x)$

In accord with the Continuation Hypothesis, quantificational rules like these have no direct counterpart in the basic grammar. That is, they only make sense as additions to a continuized grammar like that in (10). However, it is important to 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 <<e,t>,t>). (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 function, which accounts for their syntactic and semantic interchangeability; QNP denotations differ from the denotations of other NPs in that QNPs take advantage of the presence of continuations in a way that other NPs do not.

The rule in (13a) says that when used in a context in which *c* is its continuation, the value returned by the NP *everyone* is the result of quantification over all the possible individuals that might be fed to that continuation (ignoring animacy implications for simplicity). 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, the computation proceeds just as smoothly:

(14) John saw everyone.
[S [NP$_{SU}$ John] [VP [Vt saw] [NP$_{DO}$ everyone]]]
$\lambda c.\underline{\text{NP}}_{SU}(\lambda x.\underline{\text{NP}}_{DO}(\lambda y.\underline{\text{Vt}}(\lambda R(c((Ry)x)))))$
$\lambda c.\underline{\text{NP}}_{SU}(\lambda x.\underline{\text{NP}}_{DO}(\lambda y.c(\mathbf{saw}(x,y))))$
† $\lambda c.\underline{\text{NP}}_{DO}(\lambda y.c(\mathbf{saw}(\mathbf{j},y)))$
$\lambda c.((\lambda c.\forall x : c(x))(\lambda y.c(\mathbf{saw}(\mathbf{j},y))))$
$\lambda c.\forall x : c(\mathbf{saw}(\mathbf{j},x))$

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

Note that at the line marked with a dagger ('†'), lambda conversion reveals that the continuation for the direct object NP is $\lambda y.c(\mathbf{saw}(\mathbf{j}, y))$: roughly (ignoring the continuation variable c) the property of being seen by John. This property does not correspond to any syntactic constituent. It is purely a semantic object, constructed automatically by the system for keeping track of continuations. As a result, in this system there is no awkwardness or asymmetry between subject and non-subject positions, as there is in the QR account.[1]

Quantificational determiners. The rules in (13) treat *everyone* and *someone* as lexical (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 determiners—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 also reanalyzing determiners as choice functions at the same time is a heavy load; therefore, for purely expository reasons, I will provide a special composition rule for quantificational determiners:

(15)	SYNTAX	SEMANTICS
	NP $\rightarrow$ Det N	$\underline{\text{Det}}(\underline{\text{N}})$

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

(16)	every	$\lambda\underline{\text{N}}\lambda c.\underline{\text{N}}(\lambda P.\forall x : P(x) \rightarrow c(x))$
	a, some	$\lambda\underline{\text{N}}\lambda c.\underline{\text{N}}(\lambda P.\exists x : P(x)\&c(x))$
	most	$\lambda\underline{\text{N}}\lambda c.\underline{\text{N}}(\lambda P.\mathbf{most}(P, c))$
	no	$\lambda\underline{\text{N}}\lambda c.\underline{\text{N}}(\lambda P.\neg\exists x : P(x)\&c(x))$

Here, $\rightarrow$, &, and $\neg$ 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. A few examples will illustrate these definitions in action. Consider the grammar consisting of the union of the rules in (10), (13), (15), and (16). That grammar generates the following analyses:

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

Note that the interpretation in (17c) corresponds to inverse scope, i.e., the direct object takes scope over the subject. This shows that despite being an 'in situ' analysis, nothing in the continuation

[1]In general, for any syntactic constituent, the value of its continuation will be equivalent to a lambda-abstract in which the constituent in question has been replaced with a variable of the same type as that constituent. The fact that the continuations semantics automatically constructs such abstract semantic constituents for each syntactic phrase leads to new analyses of phenomena that depend on forming such abstracts, including focus constructions and perhaps certain types of genericity. See Barker ms. for details and discussion.

mechanism itself biases towards linear scope or inverse scope.

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

2.3. Scope ambiguity: a question of priority

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

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

(18) a. S $\rightarrow$ NP VP $\lambda c.\underline{\mathrm{VP}}(\lambda P.\underline{\mathrm{NP}}(\lambda x.c(P(x))))$

b. S $\rightarrow$ NP VP $\lambda c.\underline{\mathrm{NP}}(\lambda x.\underline{\mathrm{VP}}(\lambda P.c(P(x))))$

But we may just as well have used (18b). Substituting (18b) in the example grammar will allow the subject to take wide scope over the VP.

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 at the level of the un-continuized grammar (see, e.g., Meyer and Wand 1985:223). At the continuation level, the result is the same no matter what the order in which lambda conversion is carried out. That is, one of the advantages of making continuations explicit is that it allows order-of-execution to be modeled in an order-independent fashion. However, because quantificational denotations exist only at the continuation level (by hypothesis), it does not make sense to think of scope ambiguity as literally corresponding to different order of execution; nevertheless, we can still reasonably use the term PRIORITY in its non-temporal sense. 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 merely hypothesizing that quantificational elements manipulate continuations automatically predicts not only scope displacement, but scope ambiguity as well. (Section 3 discusses scope ambiguity in some detail.)

At this point we have a unified explanation for the questions in (1). As for the question of NP Duality, QNPs denote the same type of function as other NPs, which explains their syntactic interchangeability; they differ only in that QNPs exploit the presence of continuations in a way that other NPs do. As for scope displacement and scope ambiguity, merely stating the truth conditions for QNPs in terms of continuations automatically leads to scope displacement and scope ambiguity without further stipulation.

2.4. Bounding scope displacement.

This subsection and the next address two potential worries that may spring to mind immediately concerning the viability of continuations as a theory of quantification.

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

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

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 fragment in the appendix for an analysis of these sentences.)

However, QNPs cannot take scope outside of their minimal tensed S. Just as in every theory of quantifier scope, something special must be said about tensed Ss. 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 $\rightarrow$ NP VP $\lambda c.\underline{\mathrm{VP}}(\lambda P(\underline{\mathrm{NP}}(\lambda x.c(P(x)))))$
b. NEW S $\rightarrow$ NP VP $\lambda c.c(\underline{\mathrm{VP}}(\lambda P(\underline{\mathrm{NP}}(\lambda x.P(x)))))$

(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 clause's continuation, *c*, is inside the scope of the subject and of the verb phrase in the first version, but is outside in the second.

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

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 basic (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 1998:215 for a derivation of the analogous constraint in QR theories.)

Obviously, much more would have to be said to accommodate the intricacies of scope islands; all I intend to do here is show one way that such constraints can be expressed in a continuized grammar. One important case, however, is the claim that an NP is a scope island for quantificational NPs inside of it, which is discussed in section 3.

2.5. Binding

What about quantificational binding? This subsection makes a few brief remarks to suggest how standard techniques can suffice to provide appropriate truth conditions for binding pronouns in the scope of quantificational operators.

As in many theories, let the syntax associate each NP with an index according to a number of constraints. To simplify, anaphoric NPs (which include at least pronouns and reflexives) may (sometimes must) share an index with another NP (or set of NPs); otherwise, indices must be pairwise distinct. Anaphoric NPs will have denotations that depend on context. Here I'll adopt the standard practice of translating pronouns as variables in the logical translation language. The interpretation of expressions in the logical translation language, then, will be relative to a contextually-supplied function assigning variables to individuals. As in most theories, binding is accomplished by manipulating the assignment function against which subexpressions are to be evaluated. For instance, we will elaborate the rule for *everyone* given in (13) as follows:[1]

(22) $[\![\textit{everyone}_i]\!]^g = \lambda c.\forall x[\![c(x)]\!]^{g[x/i]}$

This rule guarantees that any pronoun that is coindexed with the relevant occurrence of the NP *everyone* (and that comes under the semantic scope of the NP and that is not otherwise bound by an intervening operator) will take on the same value as x for each quantificational case. For instance, translating the pronoun $\textit{she}_i$ as the variable x_i, and assuming that $[\![x_i]\!]^g = g(i)$, we have:

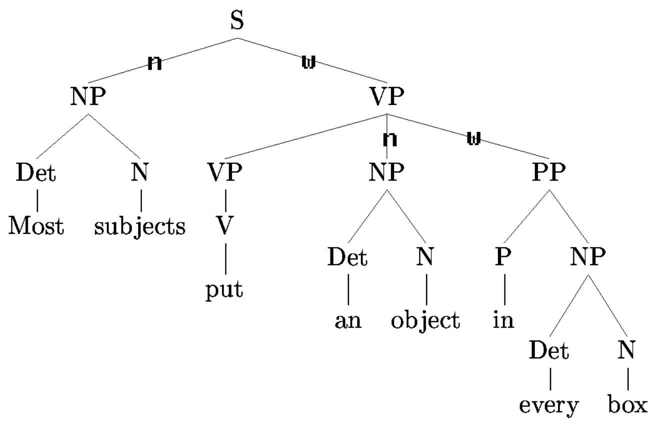
(23) $[\![\textit{everyone}_i \textit{ thinks she}_i \textit{ is intelligent}]\!]^g = \forall x[\![\mathbf{thinks}(x, \mathbf{intelligent}(x_i))]\!]^{g[x/i]}$

I assume that it is fairly obvious how to make appropriate adjustments throughout the grammar. See Barker ms. for a complete fully explicit fragment combining continuations with quantificational binding.

3. A new empirical generalization: Integrity

In order to visualize the possible scopings predicted by the continuation analysis, just mark which branches in the syntactic tree have higher priority.

[1]As usual, the denotation brackets ('$[\![\bullet]\!]$') on the left side of the equal sign and on the right side of the equal sign are distinct functions: one interprets syntactic structures of English, and the other interprets expressions in the logical translation language.

(24) 

On the labelling in (24), the subject quantifier takes narrow scope with respect to the verb phrase. Within the verb phrase, the locative NP takes scope over the direct object. (Obviously, if more than two sister branches contain quantificational elements, a more elaborate labelling system than 'W' and 'N' will have to be developed.) On the scoping determined by (24), then, *every box* takes scope over *an object*, and both take scope over *most subjects.* (This representation will remind some readers of the method for determining primary and secondary stress in metrical phonology, e.g., Liberman and Prince 1977.)

It is easy to see that in (24), the number of logically distinct scopings permitted by the continuation analysis is 4, since all we can vary is whether the subject takes scope over the verb phrase, and, within the verb phrase, whether the locative takes scope over the direct object. In contrast, many versions of QR predict 6 scopings. In general, QR provides $n!$ orderings when there are n QNPs within a clause, no matter what the syntactic configuration of the QNPs (ignoring scope islands). The continuation method predicts a factorial number of orderings only among NPs that are (embedded within) siblings (see section 5 for details). Thus unlike QR, on the continuation analysis the number of scope interpretations depends on the syntactic structure within a given clause.

The missing readings predicted possible by QR are the two readings in which the subject intervenes in scope between the direct object and the indirect object. In general, the continuation approach obeys the following constraint:

(25) **The syntactic constituent integrity scoping constraint ('Integrity')**: 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.

This constraint requires that the quantificational elements contained by a syntactic constituent form a certain kind of unit semantically as well. Specifically, quantificational elements outside of a constituent may not intervene in scope between quantificational elements within that constituent.

Whether a natural language obeys the constituent integrity constraint is an empirical matter. I conjecture that English does obey the constraint.

For example, with respect to (24), the claim is that neither of the following two scopings is grammatical:

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

As for the reading on which the indefinite gets widest scope (in (26a)), even if we found such a construal, we would have to wonder whether it arose through non-quantificational analyses of wide-scope indefinites (e.g., Kratzer 1998). Therefore I will concentrate on (28b), the scoping on which the indefinite takes narrowest scope (and presumably must therefore be quantificational), so that *every* takes scope over *most* which takes scope over *an.* The disputed truth conditions are:

(27) $\forall x.\mathbf{box}(x) \rightarrow \mathbf{most}(\mathbf{subjects}, \lambda y \exists z.\mathbf{object}(z) \& \mathbf{put}(y, z, x))$

In order to be sure whether or not (24) can have these truth conditions, we must construct a situation in which (27) is satisfied but no other scoping predicted by the continuation theory is satisfied. Therefore imagine a psychology experiment in which three subjects (S1, S2, and S3) are asked to drop an apple and a pen into any one of three boxes (B1, B2, B3). Further assume that after exactly one trial per subject, the **put** relation contains only the following triples:

(28) <S1, apple, B1> <S2, apple, B2> <S3, apple, B3>
<S1, pen, B2> <S2, pen, B3> <S3, pen, B1>

Then the truth conditions for the alleged scoping are satisfied, since for every box, most subjects placed an object in that box. Note that since there are only two objects but three boxes, it is impossible for any individual subject to have placed an object in every box.

Native speakers consistently deny that (24) can be true in such a situation. To the extent that further research confirms that analogous scopings are systematically missing, it supports the predictions of the integrity constraint.

Another relevant example that can be considered with respect to the same set of circumstances is:

(29) a. Every subject put an object in no box.
b. $\neg \exists b : \mathbf{box}(b) \& \forall s : \mathbf{subject}(s) \rightarrow (\exists o : \mathbf{object}(o) \& \mathbf{put}(s, o, b))$

If (29b) were a possible reading, (29a) should be capable of being true in the situation depicted in (28). Integrity predicts that (29a) cannot have the truth conditions described in (29b). Once again, native speakers confirm this prediction.[1]

The claim that English obeys the integrity constraint may be somewhat surprising, since several

[1]No doubt there are experienced logicians who have trained themselves to accept the truth conditions in (29b). One way to reconcile such intuitions with Integrity would be to continuize some grammar with a more flexible notion of syntactic constituency, such as many versions of Categorial Grammar and its descendents, on which *every subject put an object* can be combined as a syntactic constituent before combination with *no box*. Another possibility is that people who claim to get the non-integral reading may be resorting to higher-order continuations of the sort discussed in Barker ms.

decades of intensive research has failed to turn up such a constraint (as far as I know). I will suggest three reasons why this might be so.

First, it has only been in recent years that a clear picture has emerged of what ought to count as quantificational scope. The most dramatic example of a shift in the classification of examples as quantificational or not is the newly-emerging consensus that what used to be treated as wide-scope indefinites are best analyzed through other mechanisms. For a second kind of example, *few*, *many*, and *most* and other expressions now have analyses on which they denote choice functions that supply a mereological sum as a value (often called a 'witness set'—see, e.g., Beghelli et al. 1997). This complex individual then enters into entailments governed by the semantics of plurals and distributivity without requiring displaced scope. Factoring out cases that can be analyzed through non-quantificational mechanisms reduces the number of apparent scopings that need to be accounted for, and systematically missing scopings can start to emerge.

Second, with one exception I am aware of, every theory of quantification (including Flexible Types—see, e.g., (31c) in Hendriks 1988:113) naturally provides the full factorial range of scopings, there has been a theoretical bias to assume that the full factorial set of readings exists, at least as a default. The exception is Hobbs and Shieber's 1987 scoping algorithm, discussed below.

Third, I claim that integrity effects have in fact been noticed more than once, but never in their full generality. The relevant examples involve quantifiers embedded inside NPs. Consider first an example involving only two quantificational elements:

(30) a. No student from a foreign country was admitted.
b. $\forall s : \neg\exists c : \mathbf{student}(s) \& \mathbf{country}(c) \& \mathbf{from}(s, c) \& \mathbf{admitted}(s)$

On the most prominent reading of (30a), *no* takes scope over *a*. On the continuation analysis, the desired reading is produced automatically, with no stipulations specific to NP. (There will also be a scoping on which *a* takes wide scope over *no*, of course; see the fragment.)

However, it is well-known that examples like (30) pose 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 either to an uninterpretable structure, or at least to incorrect truth conditions.

May 1985 and Larson (in unpublished work described in Heim and Kratzer 1998:233) conclude that NP is a scope island: an NP embedded within an NP may move only as far as its containing NP. In addition to preventing unbound traces from arising, they note that this assumption makes some good predictions with respect to the observed range of scope orderings:

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

May observes that the sentence (31a) does not have the scoping indicated in (31b). Preventing *every city* from escaping from the object NP prevents the subject from intervening between the indefinite and the universal.

On the QR story, it is fairly natural to decide that NP is a scope island, since NP is generally an island for overt syntactic movement, and QR is supposed to obey the same constraints that govern syntactic movement. However, doing so creates two awkward problems 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). More seriously, it breaks quantificational binding in examples like (32).

(32) Someone from every city despises it.

As May and Larson both notice, if *every city* cannot move out of its containing NP, it cannot take semantic scope over the pronoun *it*, yet (32) clearly has a reading on which the pronoun is quantificationally dependent on the universal quantifier. On the continuation analysis, since *every* takes scope over the entire clause, it has no difficulty binding *it* (see Barker ms. for full details).

Thus 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 and a severe problem for quantificational binding.

The continuation analysis, remarkably, with no special stipulations at all gets scoping, binding, and NP-related Integrity effects right.

As a second example of a scope theory that embodies a part of Integrity, Hobbs and Shieber 1987 provide an LF-based scoping algorithm which gives (31a) 5 scopings instead of the full factorial 6. The missing reading is the one where *someone* outscopes *two* which outscopes *every*. As Hobbs and Shieber 1987:48 put it, “in general, a quantifier from elsewhere in the sentence cannot come after the quantifier associated with a head noun and before the quantifier associated with a noun phrase in the head noun’s complement.” This characterization is strikingly similar to the Integrity constraint in spirit, and I conjecture that the class of scopings ruled out for structural reasons by Hobbs and Shieber’s algorithm are a subset of the scopings ruled out by Integrity.

However, when none of the quantificational elements contains another, Hobbs and Shieber’s algorithm can deliver the full factorial range of scopings. In particular, they provide all 6 possible scopings for (24) instead of the 4 consistent with Integrity. In fact, even when one QNP contains another, Hobbs and Shieber’s algorithm generates some readings that violate Integrity, including the reading discussed above as 31b. Thus Integrity makes significantly more general (i.e., stronger) empirical claims about structurally illicit scopings than Hobbs and Shieber’s algorithm.

4. Generalized Coordination without type-shifting

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.

I assume, along with many others, that there are at least two kinds of coordination, which I will call REDUCIBLE versus LOCAL:

(33) REDUCIBLE COORDINATION:
 a. John and Mary drank a beer.
 b. John drank a beer and Mary drank a beer.

The truth conditions of one legitimate reading of (33a) can be accurately expressed by unpacking the coordination into conjoined clauses as in (33b)—in other words, the meaning of the NP conjunction can be reduced to clausal conjunction.

(34) LOCAL COORDINATION:
 a. John and Mary are a happy couple.
 b. #John is a happy couple and Mary is a happy couple.

Reducing the conjunction in (34a) to clausal coordination as in (34b), however, leads to an ill-formed paraphrase.

The truth conditions for the most natural interpretation of (34a) depends on conjoining NP meanings. Apparently, at least some conjoined NPs have denotations that involve complex individuals (mereological sums, perhaps as familiar from Link 1983). This type of conjunction is 'local' in the sense that the semantic function expressing the conjunction operates on the semantic values of the conjuncts instead of expressing the meaning of the conjunction as a function of clausal conjunction.

Lasersohn 1995 shows that properly-crafted meanings for local conjunction often produce truth conditions that are entailed by the truth conditions of the reduction analysis. This calls into question whether we need reducible conjunction in addition to local conjunction. Yet even if local conjunction will suffice for conjunction, a generalized treatment for disjunction is harder to avoid, since disjunction seems to have only a reducible interpretation in all of its uses. In addition, I will repeat below (see (43)) an argument due to Hendriks 1993 suggesting that local coordination is not sufficient even for conjunction.

In any case, let us concentrate for now on reducible coordination. Partee and Rooth 1983, building on work of Gazdar, von Stechow, and Keenan and Faltz, deal with reducible conjunction in 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 proper names (type e).

Second, they rely on a schema expressing the meaning of a coordinate structure as a function of a denotation for the conjunction.

(35) SYNTAX SEMANTICS
 $X \rightarrow X_l$ and X_r $\mathbf{and}_{\langle\alpha,\beta\rangle}(X_l, X_r)$

According to this schema, there is a distinct denotation for the conjunction for each distinct semantic type associated with the syntactic category that instantiates the metacategory X. That is, the interpretation for coordinated sentences (X = S) involves the function $\mathbf{and}_t$, the interpretation for coordinated verb phrases (X = VP) involves the function $\mathbf{and}_{\langle e,t\rangle}$, the interpretation for coordination noun phrases (X = NP) involves the function $\mathbf{and}_{\langle\langle e,t\rangle,t\rangle}$, and so on.

Third, they relate the meaning of higher-order reducible conjunction to lower-order denotations. Let L and R be meanings of type $\langle\alpha, \beta\rangle$. Then Partee and Rooth 1983 have:

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

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

The claim of such an analysis is that reducible *and* is polysemous. The schema in (36) is usually construed as a type-raising operator: we posit a single lexical meaning for *and*, namely, $\mathbf{and}_{t}$, and generate an infinite number of other denotations by repeated instantiation of (36).

Now consider one way of achieving an equivalent analysis of reducible conjunction in a continuation grammar.

(37)	SYNTAX	SEMANTICS
	$X \rightarrow X_l$ and X_r	$\lambda c.\mathbf{and}_{t}(\underline{X}_l(c), \underline{X}_r(c))$

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, do it to the right conjunct, and conjoin the resulting truth values".

Just as in (35), (37) schematizes over a range of conjoinable syntactic categories. However, there is no need to state a separate schema governing the function denoted by the conjunction—the semantic rule in (37) gives the desired result automatically. It mentions only the basic truth-value operator $\mathbf{and}_{t}$, 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., "$\underline{X}_l(c)$") is guaranteed to be a truth value, by construction. In other words, the notion of a conjoinable type is embodied in 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" is equivalent to the claim that reducible coordination lives at the level of continuations.

In sum, Partee and Rooth 1983 need a syntactic schema, a type-shifting rule, and a notion of conjoinable type. If expressions are allowed to manipulate their continuations, all that is needed is a single syntactic schema.

Some concrete examples will illustrate the use of (37). The fragment in the appendix instantiates the schema for syntactic categories S, VP, Vt, and NP:

(38)	a.	John left and John slept.	$\mathbf{and}(\mathbf{left}(\mathbf{j}), \mathbf{slept}(\mathbf{j}))$
	b.	John left and slept.	$\mathbf{and}(\mathbf{left}(\mathbf{j}), \mathbf{slept}(\mathbf{j}))$
	c.	John saw and liked Mary.	$\mathbf{and}(\mathbf{saw}(\mathbf{j}, \mathbf{m}), \mathbf{liked}(\mathbf{j}, \mathbf{m}))$
	d.	John and Mary left.	$\mathbf{and}(\mathbf{left}(\mathbf{j}), \mathbf{left}(\mathbf{m}))$

These translations use the logical constant **and** for $\mathbf{and}_t$.

Proper names can be freely coordinated with QNPs in any syntactic position. To give one scoping of just one example (an equivalent analysis is provided by the fragment):

(39) a. Tom met John and every woman.

b. $\mathbf{and}(\mathbf{met}(\mathbf{t}, \mathbf{j}), \forall z(\mathbf{woman}(z) \rightarrow \mathbf{met}(\mathbf{t}, z)))$

Reducible conjunction in a direct object gets unpacked so that (39a) comes out as equivalent to *Tom met John and Tom met every woman.*

To say that reducible coordination "lives at the level of continuations" entails that coordination interacts with quantification. For instance, the schema in (37) must also be instantiated for the category N.

(40) a. Every friend and lover left.

b. $\forall z : (\mathbf{and}(\mathbf{friend}(z), \mathbf{lover}(z))) \rightarrow \mathbf{left}(z)$

c. $\mathbf{and}((\forall z : \mathbf{friend}(z) \rightarrow \mathbf{left}(z)), (\forall z : \mathbf{lover}(z) \rightarrow \mathbf{left}(z)))$

In (40b), conjoining common nouns interacts with the quantification due to *every* in a way that does not lead to a logical translation with sentences conjoined at the highest level. The reason is that *every* quite literally takes scope over the conjunction. If we choose instead the NP composition rule that allows elements inside the NP to take wider scope, we get the translation in (40c), which is also a legitimate reading of (40a).

Partee and Rooth note that type-shifting analyses predict that coordination will interact with quantifier scope. They are skeptical that the full range of scope relationship occur, citing (41).

(41) Every student got a D or failed.

Partee and Rooth claim that (41) does not admit of the reading on which disjunction takes scope over *every*, which would entail that either every student got a D or every student failed. However, I agree with Hendriks 1988:117, 1993:94 that such readings are detectable in the right context. To his examples, I will add

(42) a. [American foreign policy during the Cold War was founded on the premise that] every nuclear nation must remain peaceful or be annihilated.

b. [When you place dominos in a circle and then toss a marble at them, then depending on where the marble lands,] every domino falls or remains standing.

I take it that (42a) has an interpretation whose truth conditions require that every nation remains peaceful or every nation is annihilated.

In addition, Hendriks 1993:99 argues that some sentences clearly have interpretations that can only be arrived at by giving coordination scope over syntactically superordinate quantifiers.

(43) a. [In those days of tight federal budgets, the best we could guarantee was that] an inspector visited every town and phoned every village.

b. $\mathbf{and}(\forall z : \mathbf{town}(z) \rightarrow \exists u : (\mathbf{inspector}(u) \& \mathbf{visited}(u, z)),$
$\quad \forall z : \mathbf{village}(z) \rightarrow \exists u : (\mathbf{inspector}(u) \& \mathbf{phoned}(u, z)))$

(The ampersands (‘&’) are due to the truth conditions for the indefinite determiner, and are simply an infix variant of $\mathbf{and}_t$.) The interpretation in (43b), of course, is one of the readings that the continuation analysis just given provides. If this is indeed a legitimate reading for (43a) (and I believe that it is), it poses a severe problem for many theories of quantifier scope, including the simplest version of the QR view. The only well-formed QR interpretations allow for a different inspector for each town, but only if each such inspector also phones every village (or vice-versa, swapping towns with villages).

Generalized coordination is often cited as the most compelling, or at least the most straightforward, motivation for recognizing type-shifting as an indispensable technique. (There are plenty of other motivations for type-shifting, of course, such as Partee 1987.) Hendriks (e.g., 1988:100) suggests that as long as we need type-shifting anyway for describing generalized coordination, why not use type-shifting to handle quantifier scope? This section turns this reasoning on its head: as long as we need continuations to handle quantification, why not use them to provide a simple semantic treatment of reducible coordination that does not depend on type-shifting?

Perhaps even more interesting than the simplicity of the continuation treatment of reducible coordination, the continuation hypothesis provides insight into why there should be two types of coordination in the first place. Local coordination lives at the basic level that ignores the presence of continuations; but as soon as we allow denotations to manipulate continuations, it is quite natural to reanalyze a coordinating particle as operating at the level of continuations.

5. The continuation schema

The relationship between a basic grammar and the equivalent continuized version can be expressed in a single schema. This is important for formalizing the sense in which continuations constitute a unitary concept, and also for proving that the size of a continuized grammar is a finite function of the size of the basic grammar.

The schema will show how to take a basic grammar G and produce an equivalent continuized grammar $\underline{G}$. The construction here is analogous to the many procedures in the computer science literature for transforming a program that does not involve continuations (i.e., a program said to be written in Direct Style) into one that does involve continuations (resulting in a program in Continuation-Passing Style). These techniques are usually called a CPS transform, and they come in many flavors depending what order of evaluation they model, whether they correspond to call-by-name, call-by-value, etc.[1]

Much of the complexity in stating the schema will come from making it as general as possible. Continuizing a grammar does not depend on the grammar containing any restricted set of syntactic or semantic operations; literally any grammar with a compositional semantics will do (and some non-compositional grammars, for that matter). PTQ can be continuized, for instance, or even a grammar that includes QR.

In order to achieve this level of generality, the schema will be framed in terms of the extension of a grammar. Any grammar G licenses a set of analyses each of which has the following form:

(44) $F, M, \langle e, l, m\rangle, \langle e_1, l_1, m_1\rangle, \ldots, \langle e_n, l_n, m_n\rangle$

The sequence in (44) expresses the fact that G associates an expression e with a syntactic category label l and a semantic value m just in case $e = F(e_1, \ldots, e_n)$ and $m = M(m_1, \ldots, m_n)$, where F is a syntactic operation specifying how the constituents $e_1 \ldots e_n$ in syntactic categories $l_1 \ldots l_n$ should be combined to form the derived expression e, and M is a semantic operation specifying how to compute a meaning m for e based on the meanings $m_1 \ldots m_n$ of e's subconstituents. (In the context-free grammars used in this paper, F is always concatenation, and M is always functional application, but other syntactic operations are possible. For instance, F could be head wrap, quantifying in, QR, etc.) Then $\underline{G}$ will be just like G except that each analysis in the form given in (44) will be replaced with $n!$ new analyses as follows: for each permutation function f (an automorphism on the first n ordinals), the following will be an analysis licensed by $\underline{G}$:

(45) $F, M_f, \langle e, l, \underline{m}\rangle, \langle e_1, l_1, \underline{m}_1\rangle, \ldots, \langle e_n, l_n, \underline{m}_n\rangle$

where M_f is determined based on M and f by the following requirement:

(46) **The Continuation Schema**:
$\underline{m} = \lambda c.\underline{m}_{f(1)}(\lambda x_{f(1)}[\ldots[\underline{m}_{f(n)}(\lambda x_{f(n)}[c(M(x_1, \ldots, x_n))])]\ldots])$

The underscores in (45) and (46) are reminders that the semantic values of the subconstituents according to $\underline{G}$ will themselves be continuized, i.e., they will denote continuation-taking functions.

[1]There is a subtle difference between the Continuation Schema here and the typical CPS transform. Typical CPS transforms (see, e.g., Meyer and Wand 1985) map a functional expression of type $\langle\alpha, \beta\rangle$ into transformed expression of type $\langle\alpha, \beta\rangle'$, where $\langle\alpha, \beta\rangle' = \langle\alpha', \langle\langle\beta', \sigma\rangle, \sigma\rangle\rangle$, and where σ is the type of an answer, i.e., the output of a computation. In the system given here, function-denoting expressions are mapped instead to transformed expressions of type $\langle\alpha, \beta\rangle' = \langle\langle\langle\alpha, \beta\rangle, \sigma\rangle, \sigma\rangle$. See Barker ms. for a more detailed comparison and discussion.

(Here and throughout the paper I use square brackets instead of parentheses purely as a visual aid to clarity.) As (46) shows, the permutation functions f determine the semantic ‘priority’ of the subconstituents.

Clearly G and $\underline{G}$ are strongly equivalent syntactically, since their syntactic components are identical. They are also strongly equivalent semantically in the following sense: if e is a syntactic structure that G associates with a meaning m, then $\underline{G}$ associates e with a meaning $\underline{m}$ such that $\underline{m}(\lambda c.c) = m$. (This theorem is analogous to Plotkin’s 1974 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.

Theorem: For any function g, $\underline{m}(\lambda c.g(c)) = g(m)$.

Proof: The proof involves recursion on the syntactic structure of e. The base case is when $n = 0$, i.e., e is a lexical item, in which case G associates e with a meaning $m = M$ where M is a constant function (a function taking zero arguments). Instantiating the continuation schema, $\underline{G}$ associates e with a meaning $\underline{m} = \lambda c.c(M)$. Therefore $\underline{m}(\lambda c.g(c)) = [\lambda c.c(M)](\lambda c.g(c)) = g(M) = g(m)$, and the claim is true. For the recursive case, assume that the claim is true for each subexpression of e. First consider an expression e with $n = 1$ (i.e., syntactic expressions containing exactly one subexpression, e.g., unary productions), where e_1 is the unique subexpression of e. Assume that G associates e_1 with the meaning m_1 and that $\underline{G}$ associates e_1 with the meaning $\underline{m}_1$. By assumption, G associates e with a meaning $m = M(m_1)$. Then there is exactly one possible choice for f (namely, the identity function, so that $f(1) = 1$), and we have

$$\underline{m} = \lambda c.\underline{m}_{f(1)}(\lambda x_{f(1)}.c(M(x_1))) = \lambda c.\underline{m}_1(\lambda x_1.c(M(x_1))).$$

Let g be an arbitrary function. Then

$$\underline{m}(\lambda c.g(c)) = [\lambda c.\underline{m}_1(\lambda x_1.c(M(x_1)))](\lambda c.g(c)) = \underline{m}_1(\lambda x_1.[\lambda c.g(c)](M(x_1))) = \underline{m}_1(\lambda x_1.g(M(x_1))).$$

Choose $g' = \lambda c.g(M(c))$. Then $\underline{m}_1(\lambda x_1.g(M(x_1))) = \underline{m}_1(\lambda x_1.g'(x_1))$. By the recursive assumption,

$$\underline{m}_1(\lambda x_1.g'(x_1)) = g'(m_1) = [\lambda c.g(M(c))](m_1) = g(M(m_1)) = g(m).$$

Thus the claim holds for $n = 1$. If $n = 2$, 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 $\underline{G}$ associates e_1 with meaning $\underline{m}_1$ and e_2 with meaning $\underline{m}_2$. Then G associates e with the meaning $m = M(m_1, m_2)$. 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

$$\underline{m} = \lambda c.\underline{m}_{f(1)}(\lambda x_{f(1)}.\underline{m}_{f(2)}(\lambda x_{f(2)}.c(M(x_1, x_2)))) = \lambda c.\underline{m}_1(\lambda x_1.\underline{m}_2(\lambda x_2.c(M(x_1, x_2)))).$$

Once again, let g be an arbitrary function. Then

$$\underline{m}(\lambda c.g(c)) = [\lambda c.\underline{m}_1(\lambda x_1.\underline{m}_2(\lambda x_2.c(M(x_1, x_2))))](\lambda c.g(c)) = \underline{m}_1(\lambda x_1.\underline{m}_2(\lambda x_2.g(M(x_1, x_2)))).$$

Choose $g' = \lambda c.g(M(x_1, c))$. Then $\underline{m} = \underline{m}_1(\lambda x_1.\underline{m}_2(\lambda x_2.g'(x_2)))$. By the recursive assumption applied to $\underline{m}_2$,

$$\underline{m} = \underline{m}_1(\lambda x_1.\underline{m}_2(\lambda x_2.g'(x_2))) = \underline{m}_1(\lambda x_1.g'(m_2)) = \underline{m}_1(\lambda x_1.g(M(x_1, m_2))).$$

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

$$\underline{m} = \underline{m}_1(\lambda x_1.g''(x_1)) = g''(m_1) = g(M(m_1, m_2)) = g(m).$$

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 $g, g', \ldots$ in a way that allows us to apply the recursive assumption from the innermost continuation out, i.e., beginning with $\underline{m}_{f(n)}, \underline{m}_{f(n-1)}, \ldots, \underline{m}_{f(1)}$. □

Corollary (Simulation): $\underline{m}(\lambda c.c) = m$. **Proof**: Set $g = (\lambda c.c)$ and the result follows immediately from the theorem. □

Note that the proof goes through no matter which permutation functions are chosen in the course of constructing the continuized meanings. This means that for every $\underline{m}$ that $\underline{G}$ associates with e, $\underline{m}(\lambda c.c) = m$.

Of course the semantic part of the equivalence between G and $\underline{G}$ does not hold in general if we add lexical items or composition rules to $\underline{G}$ 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, continuation semantics 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 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.

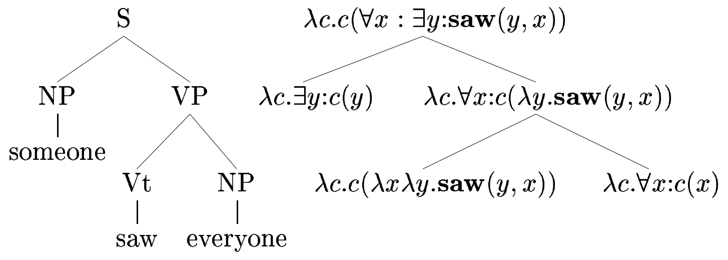
Zadrozny 1996 proves a complementary theorem: if denotations are allowed be abstract in certain ways, once again it is possible to provide a given syntax with any desired meaning relation in a compositional fashion. More specifically, Zadrozny's construction replaces normal meanings with functional meanings. 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 the meaning of an expression depends in part on its form, 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 is 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 1999:314 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.

Before turning to a more serious threat to compositionality, let me clear up a potential confusion that might give the false impression that continuations somehow circumvent compositionality. Consider for a moment how QNPs take wide scope. (It may be helpful to review the discussion of (14).) In some sense, the continuation mechanism is the opposite of storage accounts: in a storage account, QNPs are placed on a list that gets transmitted up the tree until they reach the place where they scope out. In contrast, on the continuation story, the evaluation of QNPs is never delayed; they always take their scope in situ. However, by incorporating their continuation into the calculation of their value, QNPs are able to produce a result that seems to contain information from superordinate material. Isn't this a violation of the compositionality requirement that the value of an expression depends only on its *sub*parts?

The answer to this apparent paradox, of course, is that each QNP has a perfectly well-formed denotation that does not depend on material external to the NP. It is only examining expressions that are logically equivalent (equivalent by λ-conversion) that makes it seem as if they can. For example, here is the analysis for *Someone saw everyone*, with each node in the syntactic structure associated with (an expression in the logical interpretation language that evaluates to an object that serves as) its denotation in accordance with the grammar given in (10):

(47)

S
NP
VP
someone
Vt
NP
saw
everyone
$\lambda c.c(\forall x : \exists y{:}\mathbf{saw}(y,x))$
$\lambda c.\exists y{:}c(y)$
$\lambda c.\forall x{:}c(\lambda y.\mathbf{saw}(y,x))$
$\lambda c.c(\lambda x \lambda y.\mathbf{saw}(y,x))$
$\lambda c.\forall x{:}c(x)$

This calculation clearly shows that despite the fact that the direct object takes wide scope over subject, the calculation proceeds in a compositional manner in which the denotation of a complex expression depends only on the denotations of its immediate subparts and the manner in which they are composed. There is no storage, and no use of information from ‘outside’ of an expression.

Furthermore, I will close two other popular compositionality loopholes. Through the paper, I have described meanings by giving expressions in a logical language. However, like Montague, translation into the logical language is purely an expository convenience; the grammar rules are designed to operate directly on the model-theoretic objects denoted by the logical expressions. (Though anyone who has philosophical objections to model-theoretic semantics is welcome to consider the expressions in the logical language as the denotations.)

The second, somewhat related, loophole involves rejecting structured meanings along the lines of von Stechow’s structured propositions. Given (the model-theoretic approximation of) a property or a proposition, it is not in general possible to recover the denotations of the subparts that were composed to give the result. The rules in this paper are built to work only with whatever aspects of the denotations can be determined from the outside, as it were, without direct access to the internal composition of those meanings.

Nevertheless, 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 similar conclusions. For instance, Dalrymple et al. 1997:229 allow ‘ambiguities of semantic interpretation’; this point is elaborated by Crouch 1999:48. Pelletier 1994:21 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 composition operation can be associated with more than one semantic interpretation. The resulting notion of compositionality is as follows:

(48) 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**.

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 their interface: scope ambiguity is metacompositional.

Modifying the principle of compositionality as in (48) 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 (48) does not interfere in the slightest with the degree to which compositionality imposes substantive constraints on meaning relations.

8. 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 use 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 target that the various type-shifting rules of the Flexible Type system conspire to hit. 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 about the whole 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; type-shifting analysis usually need both the shifted and the unshifted version of a meaning 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 Heim, Kamp, Groenendijk and Stokhof, among others, sentences no longer denote truth values, but update functions on contexts instead. In particular, Chierchia (1995:81) presents his dynamic logic in terms very much like what we are calling continuations (though he uses the word ‘continuation’ in an informal sense 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: “Metaphorically speaking, we add to [the interpretation of S] a hook onto which incoming information can be hung”. In fact, one leading motivation in Chierchia’s case has to do with achieving scope displacement, since incorporating the content of subsequent sentences into the sentence under evaluation can allow existential quantifiers to bind pronouns in the subsequent discourse. In any case, dynamic semantics are well-known and respectable cases of type-shifting the grammar as a whole.

At this point, it should be acknowledged that continuations are an unusally difficult concept to grasp. For the computationally inclined, it may help to work through the examples illustrating the `call/cc` operator in the Scheme specification (Kelsey et al. 1998:71 ff.), especially if a computer with a Scheme interpreter is handy. Here’s one way to get at the idea: continuations are the general form of the technique that allows all NP denotations to be expressed as generalized quantifiers. That is, entities are to generalized quantifiers as an arbitrary constituent’s basic denotation is to its continuized version.

If that still doesn’t help, there should be no difficulty whatsoever understanding precisely how a continuized grammar works, even without understanding how it was derived via the Continuation Schema. After all, the fragment in the appendix is merely a context-free grammar with normal rule-to-rule translation that relies on nothing more sophisticated than lambda abstraction and functional application. That simple, finite, context-free grammar embodies all of the main results discussed above, which I recapitulate here:

From a descriptive point of view, the continuation analysis automatically makes good predictions concerning quantificational elements inside of NPs where other accounts have considerable difficulty. Furthermore, to the extent that further research supports the descriptive accuracy of the Integrity constraint, it will provide a strong argument in favor of continuations as a robust theory of quantification.

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. (There may turn out to be compelling theoretical or empirical considerations in favor of adopting an LF-based account such as QR anyway, but if so, they weaken the force of compositionality.) 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 compositional Null-LF analysis for quantification, moreover, one that does not depend on type-shifting.

Finally, from the point of view of explanation, continuations provide a new and satisfying way of unifying several aspects of nominal quantification: merely stating the truth conditions of quantificational expressions accounts for the duality of NP meaning, scope displacement, and scope ambiguity. In addition, continuations provide an analysis of reducible coordination that is significantly simpler than other accounts, and that provides an explanation for why there are two types of coordination in the first place: local coordination lives at the basic compositional level, and reducible coordination lives at the level of first-order continuations.

Appendix: Fragment with continuations

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

(49)

S $\rightarrow$ NP VP	VP(NP)
VP $\rightarrow$ Vt NP	Vt(NP)
VP $\rightarrow$ Vc S	Vc(S)
VP $\rightarrow$ Vdt NP PP	(Vdt(NP))(PP)
NP $\rightarrow$ Det N	Det(N)
N $\rightarrow$ N PP	PP(NP)
PP $\rightarrow$ P NP	P(NP)
N $\rightarrow$ Nr PPof	Nr(PP)
PPof $\rightarrow$ of NP	NP
VP $\rightarrow$ Vi	Vi
NP $\rightarrow$ John	**j**
P $\rightarrow$ from	$\lambda x \lambda y.(y, x)$
VP $\rightarrow$ left	$\lambda x.\mathbf{left}(x)$
Vt $\rightarrow$ saw	$\lambda x \lambda y.\mathbf{saw}(y, x)$
Vc $\rightarrow$ thought	$\lambda p \lambda y.\mathbf{thought}(y, p)$
Nr $\rightarrow$ friend	$\lambda x \lambda y.\mathbf{friend}(y, x)$
Det $\rightarrow$ the	$\lambda P.\mathbf{the}(P)$
N $\rightarrow$ dog	$\lambda x.\mathbf{dog}(x)$
Vdt $\rightarrow$ put	$\lambda x \lambda y \lambda z.\mathbf{put}(z, y, x)$

In the continuized grammar below, each syntactic rule with exactly two daughters is associated with a permutation function: *lin* for linear (left-to-right) scoping priority, or *inv*, for inverse (right-to-left) scoping priority. There is one syntactic rule with three daughters, namely, the ditransitive rule for *put*. For this construction, the six permutation functions are indexed by the labels ABC, ACB, etc. These labels are for convenience only, and play no part in either the syntacic or semantic operation of the context-free grammar.

(50)

Syntax	Continuized Semantics	Scoping Priority
S $\rightarrow$ NP VP	$\lambda c.\underline{\text{NP}}(\lambda x.\underline{\text{VP}}(\lambda P.c(P(x))))$	lin
S $\rightarrow$ NP VP	$\lambda c.\underline{\text{VP}}(\lambda P.\underline{\text{NP}}(\lambda x.c(P(x))))$	inv
VP $\rightarrow$ Vt NP	$\lambda c.\underline{\text{Vt}}(\lambda R.\underline{\text{NP}}(\lambda x.c(R(x))))$	lin
VP $\rightarrow$ Vt NP	$\lambda c.\underline{\text{NP}}(\lambda x.\underline{\text{Vt}}(\lambda R.c(R(x))))$	inv
VP $\rightarrow$ Vdt NP PP	$\lambda c.\underline{\text{Vdt}}(\lambda T.\underline{\text{NP}}(\lambda x.\underline{\text{PP}}(\lambda P.c((T(x))(R)))))$	ABC
VP $\rightarrow$ Vdt NP PP	$\lambda c.\underline{\text{Vdt}}(\lambda T.\underline{\text{PP}}(\lambda R.\underline{\text{NP}}(\lambda x.c((T(x))(R)))))$	ACB
VP $\rightarrow$ Vdt NP PP	$\lambda c.\underline{\text{NP}}(\lambda x.\underline{\text{Vdt}}(\lambda T.\underline{\text{PP}}(\lambda P.c((T(x))(R)))))$	BAC
VP $\rightarrow$ Vdt NP PP	$\lambda c.\underline{\text{NP}}(\lambda x.\underline{\text{PP}}(\lambda R.\underline{\text{Vdt}}(\lambda T.c((T(x))(R)))))$	BCA
VP $\rightarrow$ Vdt NP PP	$\lambda c.\underline{\text{PP}}(\lambda P.\underline{\text{Vdt}}(\lambda T.\underline{\text{NP}}(\lambda x.c((T(x))(R)))))$	CAB
VP $\rightarrow$ Vdt NP PP	$\lambda c.\underline{\text{PP}}(\lambda P.\underline{\text{NP}}(\lambda x.\underline{\text{Vdt}}(\lambda T.c((T(x))(R)))))$	CBA
NP $\rightarrow$ Det N	$\lambda c.\underline{\text{Det}}(\lambda f.\underline{\text{N}}(\lambda P.c(f(P))))$	lin
NP $\rightarrow$ Det N	$\lambda c.\underline{\text{N}}(\lambda P.\underline{\text{Det}}(\lambda f.c(f(P))))$	inv
N $\rightarrow$ N PP	$\lambda c.\underline{\text{N}}(\lambda P.\underline{\text{PP}}(\lambda Q.c(\lambda x.P(x)\&Q(x))))$	lin
N $\rightarrow$ N PP	$\lambda c.\underline{\text{PP}}(\lambda Q.\underline{\text{N}}(\lambda P.c(\lambda x.P(x)\&Q(x))))$	inv
N $\rightarrow$ Nr PP	$\lambda c.\underline{\text{Nr}}(\lambda R.\underline{\text{PP}}(\lambda x.c(R(x))))$	lin
N $\rightarrow$ Nr PP	$\lambda c.\underline{\text{PP}}(\lambda x.\underline{\text{Nr}}(\lambda R.c(R(x))))$	inv
PP $\rightarrow$ P NP	$\lambda c.\underline{\text{P}}(\lambda R.\underline{\text{NP}}(\lambda x.c(R(x))))$	lin
PP $\rightarrow$ P NP	$\lambda c.\underline{\text{NP}}(\lambda x.\underline{\text{P}}(\lambda R.c(R(x))))$	inv
N $\rightarrow$ Nr PPof	$\lambda c.\underline{\text{Nr}}(\lambda R.\underline{\text{PPof}}(\lambda x.c(R(x))))$	lin
N $\rightarrow$ Nr PPof	$\lambda c.\underline{\text{PPof}}(\lambda x.\underline{\text{Nr}}(\lambda R.c(R(x))))$	inv
PPof $\rightarrow$ of NP	$\lambda c.\underline{\text{of}}(\lambda dummy.\underline{\text{NP}}(\lambda x.c(x)))$	lin
PPof $\rightarrow$ of NP	$\lambda c.\underline{\text{NP}}(\lambda x.\underline{\text{of}}(\lambda dummy.c(x)))$	inv
NP $\rightarrow$ John	$\lambda c.c(\mathbf{j})$	
VP $\rightarrow$ left	$\lambda c.c(\lambda x.\mathbf{left}(x))$	
P $\rightarrow$ from	$\lambda c.c(\lambda x\lambda y.\mathbf{from}(y,x))$	
Vt $\rightarrow$ saw	$\lambda c.c(\lambda x\lambda y.\mathbf{saw}(y,x))$	
Vc $\rightarrow$ thought	$\lambda c.c(\lambda p\lambda y.\mathbf{thought}(y,p))$	
Nr $\rightarrow$ friend	$\lambda c.c(\lambda x\lambda y.\mathbf{friend}(y,x))$	
Det $\rightarrow$ the	$\lambda c.c(\lambda P.\mathbf{the}(P))$	
N $\rightarrow$ dog	$\lambda c.c(\lambda x.\mathbf{dog}(x))$	
Vdt $\rightarrow$ put	$\lambda c.c(\lambda x\lambda y\lambda z.\mathbf{put}(z,y,x))$	

Determiners and choice functions. Note that the basic grammar in (49) 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, $\mathbf{the}(\lambda x.\mathbf{dog}(x))$ denotes some particular element from the set of dogs. Choice functions make appealing denotations for determiners (Egli and Heusinger 1995, Kratzer 1998, inter alia).

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 basic determiner. Here is the continuized version of the basic determiner rule with type annotations added:

(51) Det $\rightarrow$ the $\lambda c_{\text{Det}}.c_{\text{Det}}(\lambda P.\mathbf{the}(P))$

Since basic determiners are choice functions (type(Det) = <<e,t>,e>), a determiner continuation (c_{Det}) is of type <<<e,t>,e>,t>, and therefore the type of a continuized determiner is <<<<e,t>,e>,t>,t>:

(52) Det $\rightarrow$ every $\lambda c_{\text{Det}}.\forall f_{<<e,t>,e>} : c_{\text{Det}}(f_{<<e,t>,e>})$

This denoation quantifies over choice functions, rather than over individuals. This may be disorienting for readers used to the PTQ or Barwise and Cooper approach to generalized quantifiers, since there seems to be a place for only one argument, when quantificational determiners usually require two arguments (a restriction and a nuclear scope). Nevertheless, the continuized choice functions provide perfectly adequate truth conditions. For instance, adding (52) to the grammar in (50), $[\![$*John saw every man*$]\!] = \forall f : \mathbf{saw}(\mathbf{j}, f(\mathbf{man}))$, 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.

(53) a. $[\![\textit{most}]\!] = \lambda c.\exists C \in \textsc{most} : \forall f \in C : c(f)$

b. $\textsc{most} = \{C : \forall P : |P| < 2 * |\{x : \exists f \in C : x = f(P)\}|\}$

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

In any case, pursuing the choice function approach gives the cleanest, most systematic continuized grammar.

To complete the fragment, as discussed above in section 2.4, the continuized rules for [S $\rightarrow$ NP VP] must be replaced with versions that make it a scope island. In addition, we must provide rules for the lexical NPs, the quantificational determiners, and instantiate the syntactic schema for generalized conjunction for the categories S, VP, Vt, NP, and N:

(54)

S $\rightarrow$ NP VP	$\lambda c.c(\underline{\mathrm{NP}}(\lambda x.\underline{\mathrm{VP}}(\lambda P.P(x))))$	lin
S $\rightarrow$ NP VP	$\lambda c.c(\underline{\mathrm{VP}}(\lambda P.\underline{\mathrm{NP}}(\lambda x.P(x))))$	inv
NP $\rightarrow$ everyone	$\lambda c.\forall x : c(x)$	
NP $\rightarrow$ someone	$\lambda c.\exists x : c(x)$	
Det $\rightarrow$ every	$\lambda c.\forall f : c(f)$	
Det $\rightarrow$ a	$\lambda c.\exists f : c(f)$	
Det $\rightarrow$ no	$\lambda c.\neg\exists f : c(f)$	
Det $\rightarrow$ most	$\lambda c.\exists C \in \textsc{most} : \forall f \in C : c(f)$	
S $\rightarrow$ S_1 and S_2	$\lambda c.\mathbf{and}(\mathrm{S}_1(c), \mathrm{S}_2(c))$	
VP $\rightarrow$ VP_1 and VP_2	$\lambda c.\mathbf{and}(\mathrm{VP}_1(c), \mathrm{VP}_2(c))$	
Vt $\rightarrow$ Vt_1 and Vt_2	$\lambda c.\mathbf{and}(\mathrm{Vt}_1(c), \mathrm{Vt}_2(c))$	
NP $\rightarrow$ NP_1 and NP_2	$\lambda c.\mathbf{and}(\mathrm{NP}_1(c), \mathrm{NP}_2(c))$	
N $\rightarrow$ N_1 and N_2	$\lambda c.\mathbf{and}(\mathrm{N}_1(c), \mathrm{N}_2(c))$	

The result of subtracting the S rules from (50), and adding the remainder to the rules in (54) is a perfectly straightforward context-free grammar with rule-to-rule interpretation that treats quantification, scope displacement, scope ambiguity, and generalized conjunction.

References

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

Beghelli, Fillippo, Dorit Ben-Shalom, and Anna Szabolcsi. 1997. Variation, distributivity, and the illusion of branching. In Anna Szabolcsi, ed. *Ways of Scope Taking*, 29–69.

Chierchia, Gennaro. 1995. *Dynamics of Meaning*, University of Chicago Press, Chicago.

Chierchia, Gennaro and Sally McConnell-Ginet. 1990. *Introduction to Formal Semantics.* MIT Press.

Crouch, Richard. 1999. Ellipsis and Glue Languages. In Shalom Lappin and Elabbas Benmamoun, eds., *Fragments: Studies in Ellipsis and Gapping*, Oxford University Press, Oxford, 32–67.

Dalrymple, Mary, John Lamping, Fernando Pereira, and Vijay Saraswat. 1997. Quantifiers, Anaphora, and Intensionality. *Journal of Logic, Language, and Information* **6**:219-273.

Dalrymple, Mary, S.M. Shieber, and F. Pereira. 1991. Ellipsis and higher-order unification. *Linguistics and Philosophy* **14**:399–452.

Danvy, Olivier, and Carolyn A. Talcott. 1998. Introduction (to a special issue on continuations). *Higher-order and Symbolic Computation* **11.2**:115–116.

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

Egli, Urs and Klaus Heusinger, eds. 1995. *Choice functions in Natural Language Semantics.* Arbeitspapier Nr. 71. Fachgruppe Sprachwisenschaft, Universität Konstanz.

Heim, Irene and Angelika Kratzer. 1998. *Semantics in Generative Grammar*, Blackwell, Oxford.

Hendriks, Herman. 1988. Type Change in Semantics: the Scope of Quantification and Coordination, in E. Klein and J. van Benthem, eds., *Categories, Polymorphism and Unification*, ITLI, Amsterdam, 96–119.

Hendriks, Herman. 1993. *Studied Flexibility*, ILLC Dissertation Series, Amsterdam.

Hobbs, Jerry and Stuart M. Shieber. 1987. An algorithm for generating quantifier scopings. *Computational Linguistics* **13**:47–63.

Hornstein, N. 1994. An Argument for Minimalism: The Case of Antecedent-Contained Deletion. *Linguistic Inquiry* **25**:455–480.

Janssen, T. 1986. *Foundations and Applications of Montague Grammar, Part I: Philosophy, Frame-*

work, Computer Science. CWI Tract 19, Center of Mathematics and Computer Science, Amsterdam.

Kazmi, Ali and Francis Jeffry Pelletier. 1998. Is Compositionality Vacuous? *Linguistics and Philosophy* **21**:629–633.

Keenan, E. 1987. Semantic case theory. J. Groenendijk, M. Stokhof, and P. Veltmann, eds., *Proceedings of the 6th Amsterdam Colloquium*, 109–132.

Kelsey, R., W. Clinger, and Jonathan Rees, eds. 1998. The revised^5 report on the algorithmic language Scheme. *Higher-order and Symbolic Computation* **11**:7-105.

Kratzer, Angelika. 1998. Scope or Pseudoscope? Are there wide-scope indefinites? In Susan Rothstein, ed., *Events and Grammar*, Kluwer, Dordrecht, 163–196.

Lasersohn, Peter. 1995. *Plurality, Conjunction and Events*, Kluwer, Dordrecht.

Liberman, M. and A. Prince. 1977. On Stress and Linguistic Rhythm. *Linguistic Inquiry* **8.2**:249–336.

Link, G. 1983. The logical analysis of plurals and mass terms, a lattice-theoretical approach. In R. Bäuerle et al., eds. *Meaning, Use, and Interpretation of Language*, Berlin, 302–323.

May, Robert. 1985. *Logical Form: Its Structure and Derivation*, MIT Press, Cambridge, Massachusetts.

Meyer, Albert R. and Mitchell Wand. 1985. Continuation semantics in type lambda-calculi (summary). In Rohit Parikh, ed., *Logics of Programs–Proceedings*, Brooklyn: Springer-Verlag:219–224.

Montague, R. 1970. 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*, Reidel, Dordrecht, 221–42. Also in R. Thomason, ed., 1974. *Formal Philosophy: Selected Papers of Richard Montague*, 247–270.

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*, Foris, 115–143.

Partee, Barbara Hall and Mats Rooth. 1983. Generalized conjunction and type ambiguity. In Rainer Bäuerle, Christoph Schwarze, and Arnim von Stechow, eds., *Meaning, Use, and Interpretation of Language*, 361–383.

Pelletier, Francis Jeffry. 1994. The Principle of Semantic Compositionality. *Topoi* **13**:11–24.

Plotkin, Gordin D. 1975. Call-by-name, call-by-value and the λ-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.* Reidel, Dordrecht.

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

Sag, Ivan. 1976. *Deletion and Logical Form.* PhD dissertation, MIT.

Sher, Gila. 1997. Partially-ordered (branching) generalized quantifiers: a general definition. *Journal of Philosophical Logic* **26**:1–43.

Shieber, Stuart, Fernando Pereira, and Mary Dalrymple. 1996. Interactions of scope and ellipsis. *Linguistics and Philosophy* **19**:527–552. Also in Shalom Lappin and Elabbas Benmamoun, eds., *Fragments: Studies in Ellipsis and Gapping,* Oxford University Press, Oxford, 8–31.

Westerståhl, Dag. 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.