Continuations and the nature of quantification *

Chris Barker
University of California, San Diego

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

1. Introduction: the Continuation Hypothesis

This paper sets out to motivate the use of continuations as a tool for describing natural language meaning. The main result shows 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:

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

(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 odd 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 1990:116) in a way that the manifest existence of scope displacement is not.

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

Under the QR account, the best we can say in answer to the duality question in (1a) is that a generalized quantifier is what a subject NP would have to denote in order to take a verb phrase as an argument.

But why are subjects special? And why repair type-mismatches via QR, rather than, say, prohibiting QNPs in non-subject positions? There may be reasonable answers to these questions, perhaps along the lines of claiming that since QR resembles overt syntactic movement, we can use it 'for free'; my point is that the fact that these questions require answers shows that duality and scope displacement are distinct phenomena according to the QR view.

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

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

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

Therefore consider the following proposal:

(2) **The continuation hypothesis**: some linguistic expressions (in particular, QNPs) have denotations that manipulate their own continuations.

As explained in detail in the sections below, this single assumption provides answers to all three of the questions in (1): (a) QNPs denote the same kind of object as other types of NP, so there is no type clash when they occur in object position or other NP argument positions; (b) because of the nature of continuations, merely stating the truth conditions of a QNP in terms of continuations automatically guarantees that

it will have semantic scope over an entire clause—in other words, scope displacement follows directly from the semantic nature of quantification; finally, (c) scope ambiguity turns out to be a natural consequence of indeterminacy in the way in which continuations get put together in the course of composing a complex meaning. In sum, duality, scope displacement and scope ambiguity all follow from the single assumption that noun phrase meanings have access to their continuations.

2. Continuations

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

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

2.1. What is a continuation

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

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

This simple example illustrates two important aspects of continuations: (1) every meaningful subexpression has a continuation; and (2) the continuation of an expression is always relative to some larger expression containing it. Thus when *John* occurs in the sentence *John left*

yesterday, its continuation is the property $\lambda x.\mathbf{yesterday}\,\mathbf{left}\,x$; when it occurs in *Mary thought John left*, its continuation is the property $\lambda x.\mathbf{thought}(\mathbf{left}\,x)\,\mathbf{m}$; and when it occurs in the sentence *Mary or John left*, its continuation is $\lambda x.(\mathbf{left}\,\mathbf{m}) \vee (\mathbf{left}\,x)$, and so on.

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

In general, given a sub-expression A of type α embedded in an expression B of type β, the continuation of A relative to B is a function of type $\langle\alpha, \beta\rangle$ abstracting over the denotation of A that characterizes how the meaning of the whole depends on the meaning of the subexpression.

Continuations, then, are just a different way of looking at the relationship between the meaning of a complex expression and the meaning of its parts. Clearly, continuations exist independently of any framework or specific analysis, and all occurrences of expressions have continuations in any language that has a semantics. Since continuations are nothing more than a perspective, they are present whether we attend to them or not. The question under consideration, then, is not whether continuations exist—they undoubtedly do—but precisely how natural language expressions do or don't interact with them.

2.2. Continuizing a simple grammar: Deriving generalized quantifiers as a special case

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

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

I will build a continuized fragment with quantification in three main steps: first, I will present a non-quantificational grammar with a direct semantics. Then I will construct an equivalent continuized grammar. Finally, I will add elements to the continuized grammar that exploit

the presence of continuations in order to account for scope displacement and scope ambiguity.

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

(3)

Syntax	Direct Semantics
S → NP VP	⟦VP⟧(⟦NP⟧)
VP → Vt NP	⟦Vt⟧(⟦NP⟧)
NP → John	**j**
NP → Mary	**m**
VP → left	**left**
Vt → saw	**saw**

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

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

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

(5)

Direct Denotation	Semantic Type	Description
⟦S⟧	t	truth value
⟦NP⟧	e	entity
⟦VP⟧	⟨e, t⟩	property (i.e., a set of entities)
⟦Vt⟧	⟨e, ⟨e, t⟩⟩	relation over entities

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

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

(6)

Direct Variable	Type
p, q	t
x, y, z	e
P, Q	⟨e, t⟩
R, S	⟨e, ⟨e, t⟩⟩

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

Since we will be dealing in continuations, we will need new logical symbols to use as variables over continuation objects. Somewhat abusing the notational conventions of computer science discussions

of continuations, I will relate a continuation variable with its direct counterpart by drawing a line over the continuation variable:

(7)

Continuation Variable	Type
$\overline{p}, \overline{q}$	$\langle \mathtt{t}, \mathtt{t} \rangle$
$\overline{x}, \overline{y}, \overline{z}$	$\langle \mathtt{e}, \mathtt{t} \rangle$
$\overline{P}, \overline{Q}$	$\langle \langle \mathtt{e}, \mathtt{t} \rangle, \mathtt{t} \rangle$
$\overline{R}, \overline{S}$	$\langle \langle \mathtt{e}, \langle \mathtt{e}, \mathtt{t} \rangle \rangle, \mathtt{t} \rangle$

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

The goal of continuizing the grammar is to provide each expression with its own continuation as an argument. Therefore if an expression of category A with semantic type α has direct denotation $[\![\mathrm{A}]\!]$, then the continuized denotation $\{\!\{\mathrm{A}\}\!\}$ will be a function from A-continuations (type $\langle \alpha, \mathtt{t} \rangle$) to the type of the larger expression as a whole, i.e., to a truth value: type $(\{\!\{\mathrm{A}\}\!\}) = \langle \langle \alpha, \mathtt{t} \rangle, \mathtt{t} \rangle$.

(8)

Continuized Denotation	Type
$\{\!\{\mathrm{S}\}\!\}$	$\langle \langle \mathtt{t}, \mathtt{t} \rangle, \mathtt{t} \rangle$
$\{\!\{\mathrm{VP}\}\!\}$	$\langle \langle \langle \mathtt{e}, \mathtt{t} \rangle, \mathtt{t} \rangle, \mathtt{t} \rangle$
$\{\!\{\mathrm{NP}\}\!\}$	$\langle \langle \mathtt{e}, \mathtt{t} \rangle, \mathtt{t} \rangle$
$\{\!\{\mathrm{Vt}\}\!\}$	$\langle \langle \langle \mathtt{e}, \langle \mathtt{e}, \mathtt{t} \rangle \rangle, \mathtt{t} \rangle, \mathtt{t} \rangle$

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

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

(9)

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

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

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

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

(10)

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

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

(11)

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

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

(12)

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

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

Note in (8) that a continuized NP denotation is of type $\langle\langle \mathtt{e}, \mathtt{t}\rangle, \mathtt{t}\rangle$. This, of course, is the type of (an extensional) generalized quantifier

as proposed in PTQ. This is the sense in which PTQ involves a limited form of continuation-passing: saying that NPs denote generalized quantifiers amounts to saying that NPs denote functions on their own continuations.

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

2.3. The nature of quantification

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

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

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

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

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

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

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

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

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

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

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

(15)

Syntax	Semantics
NP → Det N	$\{\!\{\text{Det}\}\!\}(\{\!\{\text{N}\}\!\})$

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

(16)

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

Here, $\rightarrow$, $\wedge$, 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). Recall that $\overline{x}$ is a variable over NP continuations, and therefore is a function from individuals to truth values, which is the same type as the direct variable P. Thus these denotations take a continuized nominal denotation as an argument and return a continuized NP meaning.

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

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

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

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

2.4. Scope ambiguity: a question of priority

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

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

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

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. (Both (18a) and (18b) are instances of the general continuization schema discussed in section 5.)

How shall we interpret this state of affairs? Given the equation S = VP(NP), we can either interpret the NP as providing the continuation for the VP ("What you do with a VP is apply it as a functor to the subject"), or we can interpret the VP as providing the continuation for

the subject ("What you do with a subject is feed it as an argument to a VP"). The result is the same, in the absence of quantification—but in the presence of quantification, the two perspectives lead to different relative scopings.

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

Therefore let us say that (18a) gives the VP PRIORITY over the NP, so that quantificational elements in the VP take scope over the NP. Similarly, (18b) gives the NP priority over the VP, so that the subject takes wide scope.

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

2.5. BOUNDING SCOPE DISPLACEMENT

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

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

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

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

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

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

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

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

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

2.6. Summary of section 2

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

3. NP as a scope island

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

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

The QNP *a foreign country* is embedded within the subject NP. On the most prominent reading of (22a), *no* takes scope over *a*, resulting in the truth conditions given in (22b). On the continuation analysis, this reading is produced automatically when the quantificational determiner *a* is given priority over the nominal *from a foreign country*. There is

also a so-called inverse linking reading, as given in (22b), which arises when the nominal is given priority over the determiner. (The inverse linking reading is more prominent in other examples, such as *an apple in every basket was rotten.*)

Linear scope in examples like (22) 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. It is fairly natural to decide that NP is a scope island, since NP is an island for overt syntactic movement, and QR is generally supposed to obey the same constraints that govern syntactic movement. However, allowing a QNP to adjoin to NP creates an awkward problem for interpretation: it requires type flexibility, since the raised NP is not the right type when adjoined within NP to receive its normal interpretation (see Heim and Kratzer (1998:221)). Therefore it is to the credit of the continuation analysis that it handles linear and inverse scope for QNPs within NP without special stipulation.

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

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

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

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

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

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

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

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

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

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

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

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

4. Generalized Coordination without type-shifting or type-polymorphism

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

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

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

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

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

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

(28)

SYNTAX	SEMANTICS
$\mathrm{X} \rightarrow \mathrm{X}_l$ and X_r	$\mathbf{and}_{\langle \alpha, \beta \rangle}\ (\llbracket \mathrm{X}_l \rrbracket)(\llbracket \mathrm{X}_r \rrbracket)$

Third, they provide a recursive rule characterizing how the meaning of generalized *and* for complex semantic types relates to the meaning of *and* for simpler types. Let L and R be meanings of type $\langle \alpha, \beta \rangle$. Then Partee and Rooth (1983) have:

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

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

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

Equivalently, it is possible to think of (29) as a type-shifting rule, in which case *and* is polysemous, where each distinct homophonous version of *and* takes arguments of a single semantic type. Then there

is one basic lexical meaning for *and* (namely, the operator over truth values), and the various other senses of *and* are related to each other and ultimately to the basic lexical meaning by means of a type-shifting rule resembling (29). See Heim and Kratzer (1998:182) for a discussion of this type of approach.

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

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

(30)

Syntax	Semantics
$\mathrm{X} \rightarrow \mathrm{X}_l$ and X_r	$\lambda\overline{x}.\mathbf{and}(\{\!\{\mathrm{X}_l\}\!\}(\overline{x}))(\{\!\{\mathrm{X}_r\}\!\}(\overline{x}))$

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

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

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

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

It might seem as if having a syntactic schema like (30) amounts to the same kind of polymorphism as the Partee and Rooth account. But there is a significant difference: Partee and Rooth have both syntactic polymorphism (represented in rule (28)) and also lexical semantic type-

polymorphism (as in rule (29), or, equivalently, using semantic type-shifting). On the continuations account, there is no need to recapitulate the syntactic polymorphism in the semantics. As a result, the model-theoretic meaning attributed to *and* is simpler on the continuations account: it has a single type (namely, $\langle\langle \mathtt{t}, \mathtt{t}\rangle, \mathtt{t}\rangle$), and operates on truth values and nothing else.

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

5. The continuation schema

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

The schema will show how to take a direct grammar G and produce an equivalent continuized grammar $\overline{G}$.

Much of the complexity in stating the schema will come from making it as general as possible. Literally any grammar with a compositional semantics can be continuized—including, incidentally, a grammar that has already been continuized, leading to higher-order continuations.

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

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

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

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

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

each such f, there will be a rule in $\overline{G}$ of the following form:

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

where M^r_f satisfies the following constraint:

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

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

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

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

(32) Rule in G: $\langle E^r, C^r, M^r\rangle$

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

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

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

(33) Rule in G: $\langle E^r(e_1), C^r(c_1), M^r(m_1)\rangle$

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

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

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

(34) Rule in G: $\langle E^r(e_1, e_2), C^r(c_1, c_2), M^r(m_1, m_2)\rangle$

Rules in $\overline{G}$: $\langle E^r(e_1, e_2), C^r(c_1, c_2), M^r_f(\hat{m}_1, \hat{m}_2)\rangle$
$\langle E^r(e_1, e_2), C^r(c_1, c_2), M^r_g(\hat{m}_1, \hat{m}_2)\rangle$

where
$M^r_f(\hat{m}_1, \hat{m}_2) = \lambda\overline{x}.\hat{m}_{f(1)}(\lambda x_{f(1)}.\hat{m}_{f(2)}(\lambda x_{f(2)}.\overline{x}(M^r(x_1, x_2))))$
$M^r_g(\hat{m}_1, \hat{m}_2) = \lambda\overline{x}.\hat{m}_{g(1)}(\lambda x_{g(1)}.\hat{m}_{g(2)}(\lambda x_{g(2)}.\overline{x}(M^r(x_1, x_2))))$

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

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

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

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

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

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

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

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

Let g be an arbitrary function. Then

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

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

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

Thus the claim holds for expressions licensed by rules of arity 1. If e is licensed by a rule r of arity 2, e has exactly two immediate subconstituents, e_1 and e_2. Assume that G associates e_1 with meaning m_1 and e_2 with meaning m_2, and that $\overline{G}$ associates e_1 with meaning

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

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

Once again, let g be an arbitrary function. Then

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

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

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

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

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

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

Theorem (Simulation): $\hat{m}(\lambda x.x) = m$. **Proof**: Choose $g = (\lambda x.x)$ and the result follows immediately from the lemma. □

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

6. Compositionality

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

Is compositionality an empirical issue, or formally vacuous and therefore merely a methodological preference? A number of mathematical

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

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

Pelletier (1994) explains one way that compositionality can clearly be falsified, which establishes that compositionality is empirically substantive. If there were a natural language in which ϕ and ψ were two distinct expressions that meant the same thing, and $F(\phi)$ and $F(\psi)$ were well-formed syntactic expressions that meant different things (i.e., $[\![F(\phi)]\!] \neq [\![F(\psi)]\!]$), there simply 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.

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

The alternative is to admit, contra Montague, that there is such a thing as semantic ambiguity. A number of other researchers have come to 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 formation operation can be associated with more than one semantic interpretation. The resulting notion of compositionality is as follows:

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

In the system here, each permutation function determines one "mode of composition". This places the burden of scope ambiguity on something

that is neither syntactic, nor properly semantic, but at their interface: scope ambiguity is metacompositional.

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

7. Conclusions

The continuation approach to quantification is most similar in methodological outlook to Hendriks' Flexible Types system. Both approaches respect syntactic structure, i.e., they interpret overt syntactic structure directly without recourse to invisible manipulations at LF; neither 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 underlying generalization that the various type-shifting rules of the Flexible Type system conspire to simulate.[4] Alternatively, it is possible to view the type-shifting schemata of the Flexible Types system as a decomposition of continuations into more basic type-shifting operations. One problem with this view is that it is difficult to see how either Argument Raising or Value Raising is simpler than the continuation schema given in section 5.

I have claimed that continuations do not rely on type-shifting. Yet there is an unmistakable flavor of type-shifting throughout the continuation enterprise. One way to say it is that instead of type-shifting expressions, we are type-shifting composition rules. Perhaps an even better way to view it would be to say that it is the entire grammar as

a whole that has been shifted. I will make two points in response to this thought. First, once a grammar has been shifted to its continuized version, there is no need to use the unshifted grammar; in contrast, the point of most type-shifting analysis (e.g., Partee and Rooth (1983), Partee (1987)) is that both the shifted and the unshifted version of a meaning are needed in different situations.

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

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

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

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

From a methodological point of view, taking the principle of compositionality seriously means preferring analyses in which logical form stays as close to surface syntax as possible. Hendriks' Flexible Types system shows that there are compositional alternatives on which interpretation proceeds directly from surface syntax, as long as we are willing to introduce radical type-shifting. The continuation analysis

developed here provides a second direct-compositional analysis ('direct' in Jacobson's sense) for quantification, moreover, one that does not depend on type-shifting.

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

8. Appendix: Fragment with continuations

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

(37)

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

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

(38)

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

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

But what type will a quantificational determiner be? Since all members of a syntactic category have the same semantic type, it will be the same as the type of a continuized direct determiner, i.e., type $\langle\langle\langle\langle \mathtt{e}, \mathtt{t}\rangle, \mathtt{e}\rangle, \mathtt{t}\rangle, \mathtt{t}\rangle$.

For instance, given a variable f over choice functions of type $\langle\langle \mathtt{e}, \mathtt{t}\rangle, \mathtt{e}\rangle$, here is lexical entry of the appropriate type for *every*:

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

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

the choice-function literature cited above.) For instance,

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

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

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

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

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

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

(41)

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

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

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

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

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

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

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

The variables y, f, and x correspond to the quantificational elements *someone*, *a*, and *everyone*, respectively. As discussed with respect to example (23) in section 3, because the fragment respects the Integrity generalization, it correctly predicts that there should be four scopings instead of the factorial 6.

9. References

Barker, C.: 2001, Integrity: a syntactic constraint on quantificational scoping, in K. Megerdoomian and L.A. Bar-el (eds.), *WCCFL 20 Proceedings*, pp. 101–114. Cascadilla Press, Somerville, MA.

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

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

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

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

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

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

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

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

de Groote, P.: 2001, Continuations, type raising, and classical logic, in R. van Rooy and M. Stokhof (eds.), *Thirteenth Amsterdam Colloquium*, pp. 97–101. Institute for Logic, Language and Computation, Universiteit van Amsterdam.

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

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

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

Jacobson, P.: 1999, Towards a variable free semantics. *Linguistics and Philosophy* **22**, 117–184.

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

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

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

Kratzer, A.: 1998, Scope or Pseudoscope? Are there wide-scope indefinites?, in S. Rothstein (ed.), *Events and Grammar*, pp. 163–196. Kluwer, Dordrecht.

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

Meyer, A.R. and M. Wand: 1985, Continuation semantics in typed lambda-calculi (summary), in R. Parikh (ed.), *Logics of Programs–Proceedings*, pp. 219–224. Springer-Verlag, Brooklyn, NY.

Montague, R.: 1973, The Proper Treatment of Quantification in English, in J. Hintikka, J. Moravcsik, and P. Suppes (eds.), *Ap-*

proaches to Natural Language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics, pp. 221–42. Reidel, Dordrecht. Also in R. Thomason (ed.): 1974, *Formal Philosophy: Selected Papers of Richard Montague*, pp. 247–270. Yale University Press, New Haven, CT.

Partee, B.H.: 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*, pp. 115–143. Foris, Dordrecht.

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

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

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

Reinhart, T.: 1979, Syntactic Domains for Semantic Rules, in F. Guenthner and S.J. Schmidt (eds.), *Formal Semantics and Pragmatics for Natural Language*, Reidel, Dordrecht.

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

Shan, C-C.: 2002, A continuation semantics of interrogatives that accounts for Baker's ambiguity, in B. Jackson (ed.), *Proceedings of SALT XII: Semantics and Linguistic Theory*, Cornell University Press, Ithaca, NY.

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

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

Zadrozny, W.: 1994, From compositional to systematic semantics, *Linguistics and Philosophy* **17**, 329–342.

Notes

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

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

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