Introducing Continuations

Chris Barker ⟨barker@ucsd.edu⟩
University of California, San Diego

This working paper introduces CONTINUATIONS (a concept borrowed from computer science) as a new technique for characterizing certain aspects of the semantics of a natural language. I should emphasize at the outset that this is just an introduction, and that more a rigorous and thorough treatment is under development (see Barker (ms)). In the meantime, this paper mentions certain formal results without proving them, and describes certain new empirical generalizations without exploring them. What it *will* do is provide an explicit account of a range of familiar phenomena related to quantification, including quantifier scope ambiguity, NP as a scope island, and generalized coordination. What makes the account noteworthy is that it provides a fully and strictly compositional analysis of quantification and generalized coordination that does not rely on syntactic movement operations such as Quantifier Movement, auxiliary storage mechanisms such as Cooper Storage, or type ambiguity as in Hendriks' Flexible Types system.

1. Continuations

Continuations are a well-established technique in the theory of programming language semantics. 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 functional programming languages such as Scheme, ML, Haskell, and others.

In brief, a continuation is the entire default future of a computation. For each subexpression, its continuation is its fate.

(1) a. $(7 + ((33 + 1) * x))$

b. $(33 + 1)$ $\quad\quad$ $\lambda y.7 + (y * x)$

c. 7 $\quad\quad$ $\lambda y.y + ((33 + 1) * x)$

Within the context of the larger expression given in (1a), the fate of the number denoted by the subexpression "(33 + 1)" is that it will be multiplied by the value of x, and the result of that operation will be added to 7. Expressing this continuation using the lambda-calculus, we have $\lambda y.7 + (y * x)$. Similarly, the continuation of the constant 7 is a function that takes an integer and adds it to the product of x and the sum of 33 plus 1.

Programming languages such as Scheme allow expressions to denote functions on their own continuation. It turns out that this one device is capable of recon-

R. Hastings, B. Jackson and Z. Zvolenszky (eds), *SALT XI* 20-35, Ithaca, NY: Cornell University.

structing most common programming control structures, including `if-then-else` statements, `goto` statements, `for` loops, `while` loops, the `throw/catch` construction, `return`, etc. Obviously, there is a strong analogy between such programming control constructions and quantification in natural language; for instance, the truth conditions of the sentence *Every boy left* can be rendered roughly as 'Foreach x in **boy** {if (not (x in **left**)), then return (**false**)}'. The main idea of this paper, then, is this: if continuations can provide a unified perspective on quantification-like structures in formal languages, then perhaps they can provide useful insights into quantification in natural language.

But what exactly does it mean for an expression to denote a function on its own continuation? I will answer this question in three stages. First, I will present a simple grammar without continuations. Then I will provide an equivalent grammar with continuations. Finally, I will add rules that exploit the presence of continuations in order to provide an account of quantification.

Therefore consider the following simple, context-free, extensional grammar:

(2)

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

This grammar provides the following analyses, after lambda-conversion:

(3)

John left.	**left(j)**
John saw Mary.	**saw(j, m)**

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

(4)

SYNTACTIC CATEGORY	SEMANTIC TYPE	GLOSS ON THE SEMANTIC TYPE
S	t	truth value
NP	e	entity
VP	<e,t>	property extension (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.

(5)

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).

Adding continuations to the basic grammar in (2) requires two new elements that must be carefully distinguished: continuations, and continuized denotations.

A **continuation** is a function from a normal (uncontinuized) value to the result of the entire computation. Since we will consider only declarative statements, the result of the entire computation will always be a truth value. Therefore if an expression in syntactic category X has semantic type α, the type of the continuation c_X will be $\langle \alpha, t \rangle$. Comparing (4) with (6), then, the rule of thumb is that to get the type of a continuation, add a t.

(6)

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

In particular, note that the type of an NP continuation is <e,t>. But the basic (i.e., uncontinuized) type of a VP is also <e,t>. According to (2), after all, a VP meaning is a function from NP meanings to sentence meanings. In other words, a verb phrase meaning is an NP continuation.

A **continuized denotation** is a function on continuations. I will indicate a continuized denotation by underlining, so that $\underline{VP}$ is a continuized VP denotation. For each syntactic category X with semantic type α in the uncontinuized grammar, in the continuized grammar the corresponding continuized syntactic category $\underline{X}$ will denote a function that takes a continuation of type c_X as its sole argument. It will return whatever the continuation returns; since continuations always return truth values, the semantic type of $\underline{X}$ will be $\langle \langle \alpha, t \rangle, t \rangle$. Comparing (4) with (7), the rule of thumb is that to get the semantic type of a continuized category, add two t's:

(7)

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

With these preliminaries in place, here is one way to continuize the basic grammar in (2):

(8)

	SYNTAX	CONTINUIZED SEMANTICS
a.	S → NP VP	$\lambda c_{\text{S}}.\underline{\text{VP}}(\lambda P.\underline{\text{NP}}(\lambda x.c_{\text{S}}(P(x))))$
b.	NP → John	$\lambda c_{\text{NP}}.c_{\text{NP}}(\mathbf{j})$
c.	VP → Vt NP	$\lambda c_{\text{VP}}.\underline{\text{NP}}(\lambda x.\underline{\text{Vt}}(\lambda R.c_{\text{VP}}(R(x))))$
d.	NP → Mary	$\lambda c_{\text{NP}}.c_{\text{NP}}(\mathbf{m})$
e.	VP → left	$\lambda c_{\text{Vi}}.c_{\text{Vi}}(\lambda x.\mathbf{left}(x))$
f.	Vt → saw	$\lambda c_{\text{Vt}}.c_{\text{Vt}}(\lambda x\lambda y.\mathbf{saw}(y,x))$

Since the type of a continuation can be deduced from the types of the other expressions, I will generally omit subscripts, writing 'c' instead of 'c_{Vt}'.

An example will show the sense in which the continuized grammar is equivalent to the original uncontinuized grammar. First, note that the syntax of the continuized grammar in (8) is identical to that of the original grammar in (2).

(9)

[S [NP [John]] [VP [left]]]	Syntax
$\lambda c.\underline{\text{VP}}(\lambda P.\underline{\text{NP}}(\lambda x.c(P(x))))$	Rule (8a).
$\lambda c.\underline{\text{VP}}(\lambda P.((\lambda c.c(\mathbf{j}))(\lambda x.c(P(x)))))$	Rule (8c).
$\lambda c.\underline{\text{VP}}(\lambda P.c(P(\mathbf{j})))$	λ-conversion
$\lambda c.((\lambda c.c(\lambda x.\mathbf{left}(x)))(\lambda P.c(P(\mathbf{j}))))$	Rule (8e).
$\lambda c.c(\mathbf{left}(\mathbf{j}))$	λ-conversion

According to the continuized 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

(10)

$(\lambda c.c(\mathbf{left}(\mathbf{j})))(\lambda p.p)$	application to the 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.

It is possible to generalize the continuization method illustrated above to apply to an arbitrary compositional grammar, and to prove formally that the resulting

continuized grammar is strongly equivalent to the uncontinuized grammar, modulo application to the trivial continuation; see Barker (ms) for details.

Note in (7) 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 Montague's (1970) PTQ or Barwise and Cooper (1981).

This result bears emphasizing. The main insight in PTQ is that quantificational NPs denote functions on verb phrase meanings. In continuation terms, this is equivalent to saying that quantificational NPs denote functions on their own continuations. But although Montague in effect continuized the category NP, he left all other categories uncontinuized. As a result, Montague needed to introduce a completely separate mechanism to account for quantifier scope, namely, Quantifying In (roughly equivalent to Quantifier Raising, but in reverse). The next two sections show that continuizing across the board automatically accounts for quantifier scope displacement and quantifier scope ambiguity.

2. 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.

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

The rule in (11a) 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.

Some accounts of quantification (e.g., Heim and Kratzer (1998)) motivate a rule of Quantifier Raising as a way to repair a type clash when a quantificational NP occurs in direct object position. In the system here, however, the denotation of a quantificational NP like *everyone* is the same type as the denotation of a continuized NP in (8), namely, a function from NP continuations to truth values (type <<e,t>,t>). As a result, quantificational NPs can occur in any syntactic NP position without type clash:

(12) John saw everyone.

[S [$\underline{\text{NP}}_{SU}$ John] [VP [Vt saw] [$\underline{\text{NP}}_{DO}$ everyone]]]

$\lambda c.\underline{\text{NP}}_{SU}(\lambda x.\underline{\text{NP}}_{DO}(\lambda y.\underline{\text{Vt}}(\lambda R(c((Ry)x)))))$

$\lambda c.\underline{\text{NP}}_{SU}(\lambda x.\underline{\text{NP}}_{DO}(\lambda y.c(\mathbf{saw}(x,y))))$

$\lambda c.\underline{\text{NP}}_{DO}(\lambda y.c(\mathbf{saw}(\mathbf{j},y)))$

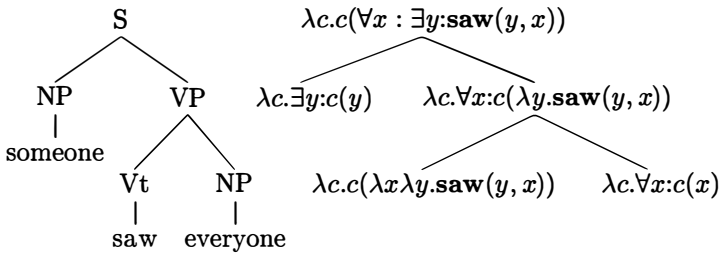
$\lambda c.((\lambda c.\forall x : c(x))(\lambda y.c(\mathbf{saw}(\mathbf{j},y))))$

$\lambda c.\forall x : c(\mathbf{saw}(\mathbf{j},x))$

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.*

Furthermore, sentences can freely contain zero, one, or more than one quantificational expression.

(13)



This calculation shows that despite the fact that the direct object takes wide scope over subject, the analysis proceeds in a strictly 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 auxiliary storage, and no use of information from 'outside' of an expression.

What about determiners? The rules in (11) treat *everyone* and *someone* as lexical (syntactically unanalyzed) NPs. Most QNPs, of course, are syntactically complex, and contain a quantificational determiner. The logical extension of the continuation strategy would be to treat uncontinuized determiners as choice functions (type <<e, t>, e>), and quantificational determiners as functions on determiner continuations (type <<<<e, t>, e>, t>, t>). This is perfectly feasible (see Barker (ms)); nevertheless, for the sake of exposition, in this paper I will treat determiners syncategorematically:

(14)

SYNTAX	SEMANTICS
NP → Det N	Det(N)

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

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

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 (8), (11), (14), and (15). That grammar generates the following analyses:

(16) 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 (16c) 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 mechanism itself biases towards linear scope or inverse scope.

Thus 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.

2.1. Bounding scope displacement.

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

(17) 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.

Assuming that this series can be extended ad infinitum and that the most natural reading of these sentences requires that *every* take wide scope over *a raindrop*, the scope of *every* must be displaced across an arbitrary number of syntactic embeddings. We need only add the most straightforward continuation of the obvious nominal modification rule:

(18) N $\rightarrow$ N PP $\lambda c.\underline{\mathbf{PP}}(\lambda Q.\underline{\mathbf{N}}(\lambda P.c(\lambda x.P(x)\&Q(x))))$

In other words, unbounded scope displacement follows automatically from the basic mechanism without needing to postulate any additional type-shifting or composition rules. (Incidentally, in standard Quantifier Raising theories, dealing with inverse

linking examples such as (17) requires stipulations such as Proper Binding, which requires that every trace left by QR must be bound in the final LF.)

However, the standard judgment is that 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:

(19) a. OLD $S \rightarrow NP\ VP$ $\lambda c.\underline{VP}(\lambda P(\underline{NP}(\lambda x.c(P(x)))))$
b. NEW $S \rightarrow NP\ VP$ $\lambda c.c(\underline{VP}(\lambda P(\underline{NP}(\lambda x.P(x)))))$

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. As pointed out by Shan (2001), Danvy and Filinski (1990) call this technique a "reset" operation.

(20) 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 (19b), all scopings of (20a) are logically equivalent to (20b). That is, *every* is not able to take scope outside of the embedded clause.

3. Scope ambiguity

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.

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

The rule in (21a) is just the (revised) S rule discussed immediately above. But we may just as well have used (21b). The difference is in the order in which the continuized NP meaning and the the continuized VP meaning contribute to the truth conditions of the sentence. Substituting (21b) 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 (21) correspond to different orders of

execution at the level of the un-continuized grammar (see, e.g., Meyer and Wand 1985:223). 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 (21a) gives the VP priority over the NP, so that quantificational elements in the VP take scope over the NP. Similarly, (21b) 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.

3.1. Integrity

There will be distinct prioritizations of continuized rules for any syntactic rule that invovles more than one subconstituent. In particular, in addition to the alternative S $\rightarrow$ NP VP rule exhibited in (21), there will be two prioritizations of the N $\rightarrow$ N PP rule, and so on. This will give rise to a considerable number of interpretations for given sentence that differ in quantifier scope possibilities; however, the range of expected scopings, though large, is significantly more constrained than under most theories of quantification. The reason is that it is a theorem that the quantifier scope possibilities allowed by continuations respects (surface) syntactic structure in a certain way:

(22) **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.

For instance, if we assume that the verb phrase in the following example constitutes a syntactic constituent, the Integrity constraint predicts that there should be four scopings, not the full factorial six predicted by most theories:

(23) Most spiders [put a foot on every lily pad].

a. most > a > every
b. most > every > a
c. every > a > most
d. a > every > most
e. a > most > every (violates Integrity)
f. every > most > a (violates Integrity)

There is at least one other theory of quantifier scope that obeys Integrity, namely, the version of combinatory categorial grammar discussed by Park (1995, 1996) and Steedman (2000). (Interestingly, the Flexible Types analysis of Hendriks (1988,

1993) does not respect Integrity.)

It is an empirical question whether the predictions of the Integrity hypothesis are accurate; I discuss this topic in some detail in Barker (2001). I will not repeat that discussion here, except to mention two factors that mitigate the empirical force of the Integrity constraint, as well as one familiar situation in which Integrity makes exactly the right prediction.

The first mitigating factor is that any scoping in which an indefinite takes anything but narrowest scope (such as the scoping in (23e)) could be due to the independent mechanism that every theory must provide for handling long-distance indefinites (a cover term including what are known as specific indefinites, de re indefinites, wide-scope indefinites, etc.). This means that potential counterexamples to Integrity that crucially rely on indefinites taking wide scope are unlikely to be compelling.

Moreover, in Barker (ms), I propose to handle long-distance indefinites by means of second-order continuations (cf. Danvy and Filinski (1990)). Second-order continuations become available when a continuized grammar is itself continuized. The basic idea is that long-distance indefinites are to quantificational indefinites as quantificational indefinites are to non-quantificational NPs. If second-order continuations are available to a grammar, they potentially provide a mechanism for describing non-Integral scopings, if necessary.

The second factor is that it cannot be taken for granted that there is only one syntactic analysis of (23). Park (1995, 1996) and Steedman (2000) provide all six sets of truth conditions for (23) by providing in effect six distinct syntactic analyses. Their alternative analyses arise through the interaction of the type-shifting operations of functional composition and type lifting. One of the main points of my paper is that type ambiguity is not required in order to give a basic account of quantification and quantifier scope; but type ambiguity may still be necessary to account for other phenomena. In particular, consider so-called non-constituent coordination:

(24) a. John ate [rice yesterday] and [beans today].
b. Mary gave [a book to John] and [a record to Bill].

The categorial grammar analyses of Dowty (1988) and others account for such examples by providing a way for the NP *rice* to compose with the adverbial *yesterday* before combining with the transitive verb *ate*. If we need composition to account for non-constituent coordination (as I suspect we do), then composition immediately provides a mechanism for constructing the disputed scopings of (23) along the lines explored by Park and Steedman.

In any case, it should be pointed out that Integrity makes excellent predictions in one domain where a number of other theories require awkward stipulations, namely, quantification within and across NPs. 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 (via Quantifier Raising) only as far as its containing NP.

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

May observes that the sentence (25a) does not have the scoping indicated in (25b). Preventing *every city* from escaping from the object NP prevents the subject from intervening between the indefinite (with respect to scope) 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 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).

On the continuation analysis, there is no need to say anything special in order for facts like (25) to fall out, since it is just a special case of Integrity.

Of course, if scopings that seem to violate Integrity can arise through functional composition, then whatever principles constrain such composition must explain why composition cannot compose parts of NPs with constituents outside of the NP. This is undoubtedly necessary, however, in order to prevent ungrammatical instances of non-constituent coordination such as **John offered every [hot dish to Mary] and [salty dish to Bill]*.

5. A payoff: generalized coordination without type ambiguity

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:

(26) 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 (26a) can be accurately expressed by unpacking the coordination into conjoined clauses as in (26b)—in other words, the meaning of the NP conjunction can be reduced to clausal conjunction.

(27) 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 (27a) to clausal coordination as in (27b), however, leads to an ill-formed paraphrase.

The truth conditions for the most natural interpretation of (27a) depends on

conjoining NP meanings. Thus 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.

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 the lexical type assigned to 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.

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

According to this schema, there is a potentially different conjunction meaning for each conjoinable category. 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:

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

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 (29) 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 (29).

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

(30) SYNTAX SEMANTICS
$X \rightarrow X_l$ and X_r $\quad \lambda c.\mathbf{and}_t(\underline{X}_l(c), \underline{X}_r(c))$

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 (28), (30) 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 (30) 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., "$\underline{\mathrm{X}}_t(c)$") 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. In a continuized grammar, all that is needed is a single schema.

Some concrete examples will illustrate the use of (30). Instantiating the schema for syntactic categories S, VP, Vt, and NP, we have:

(31) 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**(**j**, **m**), **liked**(**j**, **m**))
d. John and Mary left. **and**(**left**(**j**), **left**(**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:

(32) a. Tom met John and every woman.
b. **and**(**met**(**t**, **j**), $\forall z$(**woman**(z) $\rightarrow$ **met**(**t**, z)))

Reducible conjunction in a direct object gets unpacked so that (32a) comes out as equivalent to *Tom met John and Tom met every woman.*

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), or the approach to non-constituent coordination described above.) 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 ambiguity?

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 coor-

dinating particle as operating at the level of continuations.

6. 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.

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 keep any of the unshifted denotations; typically type-shifting analysis usually need both the shifted and the unshifted version of a meaning in different situations. In other words, even if type-shifting is used to construct the continuized grammar, once the grammar has been constructed, that grammar does not depend in any way on type ambiguity.

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. As a second example, Shan (2001) argues that the correspondence of an extensional grammar with its intensional counterpart is a shift similar to continuization (more specifically, both transforms are instances of monads).

In summary, continuations provide a new and satisfying way of unifying sev-

eral aspects of nominal quantification: merely stating the truth conditions of quantificational expressions in terms of continuations not only dervies the generalized-quantifier conception of NP meaning as a special case of a more general mechanism, it also automatically accounts for 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.

Endnotes

*Thanks to Mark Gawron, Ken Shan, Mark Steedman, and the SALT 11 audience for comments.

References

Barker, Chris. 2001. Integrity; A syntactic constraint on quantifier scope ambiguity. In *Proceedings of WCCFL 20.* Cascadilla Press.

Barker, Chris. ms. Continuations. UCSD manuscript.

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

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

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

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

Dowty, David. 1988. Type Raising, Functional Composition, and Non-Constituent Coordination. In Richard T. Oehrle, Emmon Bach, and Deidre Wheeler, eds. *Categorial Grammar and Natural Language Structures.* Kluwer, Dordrecht. 153–198.

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.

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.

Park, Jong Cheol. 1995. Quantifier Scope and Constituency. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL).* Cambridge, MA, 205–212.

Park, Jong Cheol. 1996. Quantifier Scope, Lexical Semantics, and Surface Structure Constituency. IRCS Report 96-28.

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.

Reynolds, John C. 1993. The Discoveries of Continuations. *Lisp and Symbolic 2Computation* **6**:233–247.

Shan, Chung-Chieh. 2001. Monads for natural language semantics. In K. Striegnitz, ed. *Proceedings of the ESSLLI-2001 Student Session*, 13th European Summer School in Logic, Language and Information, 285-298.

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