TYPES AS GRAPHS: CONTINUATIONS IN TYPE LOGICAL GRAMMAR

CHRIS BARKER AND CHUNG-CHIEH SHAN

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

1. Introduction

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

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

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

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

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

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

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

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

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

2. Quantification in Type Logical Grammar

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

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

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

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

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

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

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

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

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

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

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

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

3. Fusion types as trees

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

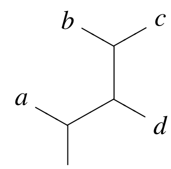
On one level, the interpretation of our system in geometric terms serves to explain the mechanics of our multimodal implementation of q. On another level, the natural geometrical interpretation that the TLG analysis lends itself to provides new insight into the logic of continuation.

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

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

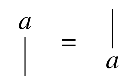
can be drawn as follows.

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



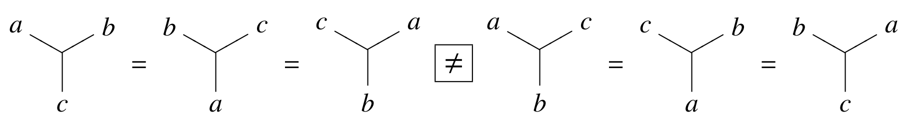
In our graphs, edges are unoriented, so the following two graphs are identical:

(5)



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

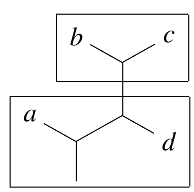
(6)



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

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

(7)

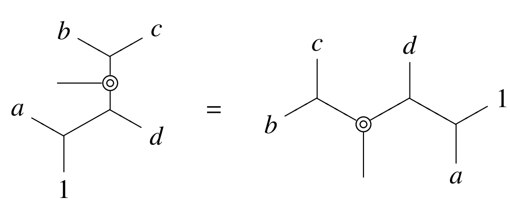


$$\iff b \circ c = B$$

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

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

(8)



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

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

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

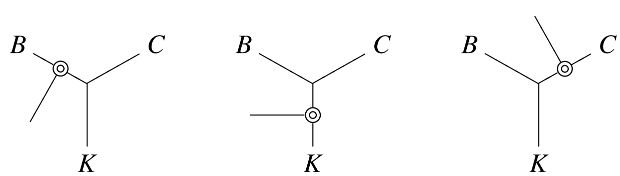
The continuation mode $\circledcirc$ treats the decomposition $A = \mathcal{K}[B]$ as a structure in its own right: $B \circledcirc K$ decomposes A into the subexpression B and the context $\mathcal{K}$.[1] In the trivial case, the graph is ripped apart at the root edge into two sides: the null context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

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

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

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

(10)



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

---

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

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

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

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

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

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

4. Scope-taking as residuation on contexts and subexpressions

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

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

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

$$\begin{array}{l} B \circ C \xrightarrow{\text{Pop/Push}} (B \circ C) \circledcirc 1 \xrightarrow{\text{Left}} B \circledcirc (C \circ 1) \xrightarrow{\text{Pop/Push}} B \circledcirc ((C \circ 1) \circledcirc 1) \xrightarrow{\text{Left}} B \circledcirc (C \circledcirc (1 \circ 1)) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \parallel \\ C \circ B \xrightarrow{\text{Pop/Push}} (C \circ B) \circledcirc 1 \xrightarrow{\text{Right}} B \circledcirc (1 \circ C) \xrightarrow{\text{Pop/Push}} B \circledcirc ((1 \circ C) \circledcirc 1) \xrightarrow{\text{Right}} B \circledcirc (C \circledcirc (1 \circ 1)) \end{array}$$

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

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

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

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

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

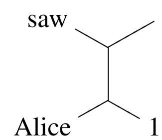
where $\triangleleft$ and $\triangleright$ are two new binary modes. Informally speaking, we distinguish between "left contexts" ($\triangleleft$) and "right contexts" ($\triangleright$) to rule out the adjacent application of Left and Right above.

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

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

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

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

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


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

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

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

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

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

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

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

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

(18)
$$\dfrac{\text{someone} \vdash s/\!\!/(np\backslash\!\!\backslash s) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{everyone} \vdash s/\!\!/(np\backslash\!\!\backslash s) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{np \circ (\text{saw} \circ np) \vdash s}\,(14)}{(np \circ (\text{saw} \circ np)) \circledcirc 1 \vdash s}\,\text{Push}}{(\text{saw} \circ np) \circledcirc (1 \circ np) \vdash s}\,\text{Right}}{np \circledcirc ((1 \circ np) \circ \text{saw}) \vdash s}\,\text{Right}}{(1 \circ np) \circ \text{saw} \vdash np\backslash\!\!\backslash s}\,\backslash\!\!\backslash\text{I}}{\text{everyone} \circledcirc ((1 \circ np) \circ \text{saw}) \vdash s}\,/\!\!/\text{E}}{(\text{saw} \circ \text{everyone}) \circledcirc (1 \circ np) \vdash s}\,\text{Right}}{(np \circ (\text{saw} \circ \text{everyone})) \circledcirc 1 \vdash s}\,\text{Right}}{np \circledcirc ((\text{saw} \circ \text{everyone}) \circ 1) \vdash s}\,\text{Left}}{(\text{saw} \circ \text{everyone}) \circ 1 \vdash np\backslash\!\!\backslash s}\,\backslash\!\!\backslash\text{I}}{\dfrac{\dfrac{\text{someone} \circledcirc ((\text{saw} \circ \text{everyone}) \circ 1) \vdash s}{(\text{someone} \circ (\text{saw} \circ \text{everyone})) \circledcirc 1 \vdash s}\,\text{Left}}{\text{someone} \circ (\text{saw} \circ \text{everyone}) \vdash s}\,\text{Pop}}\,/\!\!/\text{E}$$

(19)
$$\dfrac{\text{everyone} \vdash s/\!\!/(np\backslash\!\!\backslash s) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{someone} \vdash s/\!\!/(np\backslash\!\!\backslash s) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{np \circ (\text{saw} \circ np) \vdash s}\,(14)}{(np \circ (\text{saw} \circ np)) \circledcirc 1 \vdash s}\,\text{Push}}{np \circledcirc ((\text{saw} \circ np) \circ 1) \vdash s}\,\text{Left}}{(\text{saw} \circ np) \circ 1 \vdash np\backslash\!\!\backslash s}\,\backslash\!\!\backslash\text{I}}{\text{someone} \circledcirc ((\text{saw} \circ np) \circ 1) \vdash s}\,/\!\!/\text{E}}{(\text{someone} \circ (\text{saw} \circ np)) \circledcirc 1 \vdash s}\,\text{Left}}{(\text{saw} \circ np) \circledcirc (1 \circ \text{someone}) \vdash s}\,\text{Right}}{np \circledcirc ((1 \circ \text{someone}) \circ \text{saw}) \vdash s}\,\text{Right}}{(1 \circ \text{someone}) \circ \text{saw} \vdash np\backslash\!\!\backslash s}\,\backslash\!\!\backslash\text{I}}{\dfrac{\dfrac{\dfrac{\text{everyone} \circledcirc ((1 \circ \text{someone}) \circ \text{saw}) \vdash s}{(\text{saw} \circ \text{everyone}) \circledcirc (1 \circ \text{someone}) \vdash s}\,\text{Right}}{(\text{someone} \circ (\text{saw} \circ \text{everyone})) \circledcirc 1 \vdash s}\,\text{Right}}{\text{someone} \circ (\text{saw} \circ \text{everyone}) \vdash s}\,\text{Pop}}\,/\!\!/\text{E}$$

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

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

5. Empty antecedents

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

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

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

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

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

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

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

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

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

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

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

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

6. Refining notions of context

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

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

  In (a) and (b) above, the type K represents the outer context $\mathcal{K}$.

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

6.1. **Restrictors.**

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

---

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

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

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

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

(24) $$(s \sslash (np \bbslash s))/(np\backslash_n s).$$

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

(25)
$$
\dfrac{
\dfrac{\dfrac{\dfrac{\text{(every} \circ \text{school)} \circledcirc \text{(griped} \circ 1) \vdash s}{}}{}}{}
}{}
$$

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

If we add the lexical items

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

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

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

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

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

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

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

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

(30) Every dean of a school griped.

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

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

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

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

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

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

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

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

is not derivable; only

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

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

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

is derivable.

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

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

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

or, written equivalently,

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

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

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

and

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

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

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

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

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

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

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

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

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

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

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

(42) Unchanged case:
a. A “hole to the left” $[\ ] \circ C$, embedded in some outer context $\mathcal{K}$.
Tightened case:
b. A “hole to the right” $\Diamond B \circ [\ ]$, embedded in some outer context $\mathcal{K}$.
Unchanged case:
c. The null context $[\ ]$, the ground case for recursion.

In terms of structural postulates, we replace the Right postulate from (11) with a more specific instance.

(43) $B \circledcirc (C \circ K) \dashv\vdash (B \circ C) \circledcirc K$ (Left) $\qquad (\Diamond B \circ C) \circledcirc K \dashv\vdash C \circledcirc (K \circ \Diamond B)$ (Right)

With these changes to the grammar, only the linear-scope reading for *Someone saw everyone* (of type $\Diamond s$) remains derivable. The proof is shaped exactly as in (18), except the T and K′ postulates above are used after (14) to prove $\Diamond np \circ (\Diamond \text{saw} \circ np) \vdash \Diamond s$.[7]

(44)
$$
\begin{prooftree}
\AxiomC{$\text{someone} \vdash \Diamond s \mathbin{/\!\!/} (np \mathbin{\backslash\!\!\backslash} \Diamond s)$}
\AxiomC{$\text{everyone} \vdash \Diamond s \mathbin{/\!\!/} (np \mathbin{\backslash\!\!\backslash} \Diamond s)$}
\AxiomC{}
\RightLabel{(14)}
\UnaryInfC{$np \circ (\text{saw} \circ np) \vdash s$}
\RightLabel{$\Diamond$I}
\UnaryInfC{$\Diamond(np \circ (\text{saw} \circ np)) \vdash \Diamond s$}
\RightLabel{K′}
\UnaryInfC{$\Diamond np \circ \Diamond(\text{saw} \circ np) \vdash \Diamond s$}
\RightLabel{K′}
\UnaryInfC{$\Diamond np \circ (\Diamond \text{saw} \circ \Diamond np) \vdash \Diamond s$}
\RightLabel{T}
\UnaryInfC{$\Diamond np \circ (\Diamond \text{saw} \circ np) \vdash \Diamond s$}
\RightLabel{Push}
\UnaryInfC{$(\Diamond np \circ (\Diamond \text{saw} \circ np)) \circledcirc 1 \vdash \Diamond s$}
\RightLabel{Right}
\UnaryInfC{$(\Diamond \text{saw} \circ np) \circledcirc (1 \circ \Diamond np) \vdash \Diamond s$}
\RightLabel{Right}
\UnaryInfC{$np \circledcirc ((1 \circ \Diamond np) \circ \Diamond \text{saw}) \vdash \Diamond s$}
\RightLabel{$\mathbin{\backslash\!\!\backslash}$I}
\UnaryInfC{$(1 \circ \Diamond np) \circ \Diamond \text{saw} \vdash np \mathbin{\backslash\!\!\backslash} \Diamond s$}
\RightLabel{$\mathbin{/\!\!/}$E}
\BinaryInfC{$\text{everyone} \circledcirc ((1 \circ \Diamond np) \circ \Diamond \text{saw}) \vdash \Diamond s$}
\RightLabel{Right}
\UnaryInfC{$(\Diamond \text{saw} \circ \text{everyone}) \circledcirc (1 \circ \Diamond np) \vdash \Diamond s$}
\RightLabel{Right}
\UnaryInfC{$(\Diamond np \circ (\Diamond \text{saw} \circ \text{everyone})) \circledcirc 1 \vdash \Diamond s$}
\RightLabel{T}
\UnaryInfC{$(np \circ (\Diamond \text{saw} \circ \text{everyone})) \circledcirc 1 \vdash \Diamond s$}
\RightLabel{T}
\UnaryInfC{$(np \circ (\text{saw} \circ \text{everyone})) \circledcirc 1 \vdash \Diamond s$}
\RightLabel{Left}
\UnaryInfC{$np \circledcirc ((\text{saw} \circ \text{everyone}) \circledcirc 1) \vdash \Diamond s$}
\RightLabel{$\mathbin{\backslash\!\!\backslash}$I}
\UnaryInfC{$(\text{saw} \circ \text{everyone}) \circledcirc 1 \vdash np \mathbin{\backslash\!\!\backslash} \Diamond s$}
\RightLabel{$\mathbin{/\!\!/}$E}
\BinaryInfC{$\text{someone} \circledcirc ((\text{saw} \circ \text{everyone}) \circledcirc 1) \vdash \Diamond s$}
\RightLabel{Left}
\UnaryInfC{$(\text{someone} \circ (\text{saw} \circ \text{everyone})) \circledcirc 1 \vdash \Diamond s$}
\RightLabel{Pop}
\UnaryInfC{$\text{someone} \circ (\text{saw} \circ \text{everyone}) \vdash \Diamond s$}
\end{prooftree}
$$

The inverse-scope derivation (19) is no longer valid because the Right postulate, restricted in (43), does not apply since *someone* is impure.

What we have just seen is that a linguistic preference for linear scope reflects a computational preference for left-to-right evaluation. Of course, in many languages (including

[7]See Bernardi’s thesis (2002, page 50) for the $\Diamond$I rule used in this derivation, as well as the other natural-deduction rules for the unary operators $\Diamond$ and $\Box^{\downarrow}$, namely $\Diamond$E, $\Box^{\downarrow}$I, and $\Box^{\downarrow}$E.

English), inverse scope is available. That does not mean that order-of-evaluation effects are absent—it only means that, if they are present, they are more subtle. In the rest of this section, we examine one way to reintroduce inverse scope that gives rise to favorable empirical consequences in §7 below.

The basic idea is to treat inverse scope as an instance of *multistage programming* (see Taha and Nielsen 2003 and references therein). A multistage program is a program that generates another program (and then runs it, usually). The generating program is said to execute in an *earlier* or *outer* stage, and the generated program is said to execute in a *later* or *inner* stage. (More than two stages are also possible.) Evaluation in each stage is ordered separately: later-stage code runs only after earlier-stage code finishes generating it. Informally speaking, then, we can treat the inverse-scope reading of *Someone saw everyone* as the following outer-stage program.

(45) Run the program consisting of the word *someone*, the word *saw*, and everyone.

Italics is significant here: The sentence above only uses one quantifier (*everyone*). It also mentions a word (*someone*) that is a quantifier, but does not use it. If the domain of people under discussion is Alice, Bob, and Carol, then (45) conjoins the meanings of the sentences *Someone saw Alice*, *Someone saw Bob*, and *Someone saw Carol*. These three sentences are the inner-stage programs.

A typical multistage programming language provides facilities for: creating programs (by *quotation* or *staging*); combining programs (by concatenation); and running programs (by *unquotation* or *evaluation*). The computational intuition behind unquotation is that a quoted value is an inactive piece of program text that does not execute until the "wrapping" $\Diamond$ is removed. Removing the wrapping turns inactive data into active code. We call this step "unquotation" despite the fact that "evalution" or "eval" is the commonly used term for the same concept in the staged programming literature, to avoid confusion with the concept of evaluation order just discussed.

The T and K′ postulates in (40) and (41) above are our facilities for quoting and concatenating programs, respectively. To run programs, we add the following Unquote rule.[8]

(46) $$\Diamond\Diamond_u A \vdash \Diamond_u A \quad \text{(Unquote)}.$$

This rule makes sense because the type of a quoted program is enclosed in a $\Diamond$, and to run a program is to remove that $\Diamond$. Although quotation applies freely (using T), unquotation does not. Rather, unquotation is restricted to types of the form $\Diamond_u A$. Here $\Diamond_u$ is a unary modality that marks those types that are allowed as top-level types of quoted programs; this modality can be thought of as a feature checked by the Unquote rule. In particular, we hereafter take the clause type s to be shorthand for $\Diamond_u s'$, so that we can derive it from the quoted clause type $\Diamond s$ (shorthand for $\Diamond\Diamond_u s'$) using the Unquote rule. Here s' is an atomic formula distinct from s.[9]

In the presence of Unquote, the inverse-scope reading of *Someone saw everyone* is once again derivable, as follows. (Because s and $\Diamond s$ entail each other in the presence of

[8]In view of Moot's PSPACE-completeness result for NL$\Diamond_{\mathcal{R}^-}$ (2002, §9.2), the K′ and Unquote postulates present computational difficulties for practical parsing applications because they are *expanding*. As it turns out, we can avoid expanding rules using the same technique as in footnote 3 on page 10: add a unary mode $\Diamond_i$ (where i stands for "impure"), and put $\Diamond_i$ around every subformula not surrounded by $\Diamond$.

[9]In this paper, s is the only type that can be unquoted, that is, the only type of the form $\Diamond_u A$. A treatment of polarity licensing that is less simplistic than the one in §7.3 calls for multiple clause types that can be unquoted (Shan 2004).

Unquote, we can restore the lexical types of *someone* and *everyone* from $\Diamond s\mathbin{/\!\!/}(np\mathbin{\backslash\!\!\backslash}\Diamond s)$ back to $s\mathbin{/\!\!/}(np\mathbin{\backslash\!\!\backslash}s)$.)

(47)

$$
\begin{prooftree}
\AxiomC{$\text{everyone} \vdash s\mathbin{/\!\!/}(np\mathbin{\backslash\!\!\backslash}s)$}
\AxiomC{$\text{someone} \vdash s\mathbin{/\!\!/}(np\mathbin{\backslash\!\!\backslash}s)$}
\AxiomC{}
\RightLabel{(14)}
\UnaryInfC{$np \circ (\text{saw} \circ np) \vdash s$}
\RightLabel{Push}
\UnaryInfC{$(np \circ (\text{saw} \circ np)) \circledcirc 1 \vdash s$}
\RightLabel{Left}
\UnaryInfC{$np \circledcirc ((\text{saw} \circ np) \circ 1) \vdash s$}
\RightLabel{$\backslash\!\!\backslash$I}
\UnaryInfC{$(\text{saw} \circ np) \circ 1 \vdash np\mathbin{\backslash\!\!\backslash}s$}
\RightLabel{$/\!\!/$E}
\BinaryInfC{$\text{someone} \circledcirc ((\text{saw} \circ np) \circ 1) \vdash s$}
\RightLabel{Left}
\UnaryInfC{$(\text{someone} \circ (\text{saw} \circ np)) \circledcirc 1 \vdash s$}
\RightLabel{Pop}
\UnaryInfC{$\text{someone} \circ (\text{saw} \circ np) \vdash s$}
\RightLabel{$\Diamond$I}
\UnaryInfC{$\Diamond(\text{someone} \circ (\text{saw} \circ np)) \vdash \Diamond s$}
\AxiomC{}
\RightLabel{Id}
\UnaryInfC{$s \vdash s$}
\RightLabel{Unquote}
\UnaryInfC{$\Diamond s \vdash s$}
\RightLabel{$\Diamond$E}
\BinaryInfC{$\Diamond(\text{someone} \circ (\text{saw} \circ np)) \vdash s$}
\RightLabel{K$'$}
\UnaryInfC{$\Diamond\text{someone} \circ \Diamond(\text{saw} \circ np) \vdash s$}
\RightLabel{K$'$}
\UnaryInfC{$\Diamond\text{someone} \circ (\Diamond\text{saw} \circ \Diamond np) \vdash s$}
\RightLabel{T}
\UnaryInfC{$\Diamond\text{someone} \circ (\Diamond\text{saw} \circ np) \vdash s$}
\RightLabel{Push}
\UnaryInfC{$(\Diamond\text{someone} \circ (\Diamond\text{saw} \circ np)) \circledcirc 1 \vdash s$}
\RightLabel{Right}
\UnaryInfC{$(\Diamond\text{saw} \circ np) \circledcirc (1 \circ \Diamond\text{someone}) \vdash s$}
\RightLabel{Right}
\UnaryInfC{$np \circledcirc ((1 \circ \Diamond\text{someone}) \circ \Diamond\text{saw}) \vdash s$}
\RightLabel{$\backslash\!\!\backslash$I}
\UnaryInfC{$(1 \circ \Diamond\text{someone}) \circ \Diamond\text{saw} \vdash np\mathbin{\backslash\!\!\backslash}s$}
\RightLabel{$/\!\!/$E}
\BinaryInfC{$\text{everyone} \circledcirc ((1 \circ \Diamond\text{someone}) \circ \Diamond\text{saw}) \vdash s$}
\RightLabel{Right}
\UnaryInfC{$(\Diamond\text{saw} \circ \text{everyone}) \circledcirc (1 \circ \Diamond\text{someone}) \vdash s$}
\RightLabel{Right}
\UnaryInfC{$(\Diamond\text{someone} \circ (\Diamond\text{saw} \circ \text{everyone})) \circledcirc 1 \vdash s$}
\RightLabel{Pop}
\UnaryInfC{$\Diamond\text{someone} \circ (\Diamond\text{saw} \circ \text{everyone}) \vdash s$}
\RightLabel{T}
\UnaryInfC{$\text{someone} \circ (\Diamond\text{saw} \circ \text{everyone}) \vdash s$}
\RightLabel{T}
\UnaryInfC{$\text{someone} \circ (\text{saw} \circ \text{everyone}) \vdash s$}
\end{prooftree}
$$

As noted above, the Unquote rule is necessary to derive this reading if the Right rule is restricted as in (43). More precisely, in order for one quantifier to take inverse scope over another, the Unquote rule must apply to the clause produced by the narrower-scope quantifier (here *someone*).

The identity types (the 1's) and the unary modes add considerable complexity to the basic analysis of *Someone saw everyone* given above in (18). The payoff, as we shall see in the next section, is that these complications allow control over order of evaluation, which provides an account of fairly subtle and complex linguistic phenomena.

7. Beyond quantification

In this section, we survey continuation-based analyses of several linguistic phenomena, and sketch how those analyses can be transliterated into the TLG approach developed in this paper. The phenomena discussed are quite intricate, and we cannot hope to be complete or comprehensive here—many additional details are available in the papers cited. Rather, our main purpose is to show how continuations can offer explanatory insights previously unavailable in TLG, as well as improve the empirical coverage as compared to previous TLG accounts of the same phenomena.

7.1. **Binding and crossover.** In another manuscript (Shan and Barker 2003), we argue that order of evaluation provides a unified explanation for two distinct phenomena in English, weak crossover and superiority. We will compare our approach to Jäger's (2001) TLG analysis of weak crossover. In this case, the new explanation is the claim that weak crossover and superiority stem from the same underlying mechanism (namely, uniform default left-to-right evalaution). The improved empirical coverage concerns the interaction of pied-piping, *wh*-binding, and weak crossover.

Weak crossover is the name for the fact that a quantificational NP must usually precede any pronoun that it binds:

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

We claim that weak crossover follows from the assumption that by default, people evaluate expressions from left to right, combined with the fact that a quantifier must be evaluated before evaluting any pronoun that it binds.

Variable binding, of course, is a prototypical side-effect in computer programming languages. Since continuations model side-effects, binding is an ideal arena in which to test the utility of continuations for describing natural language.

Binding in general poses a problem for TLG. Standard multimodal TLG is linear, but binding by its very nature involves using a resource more than once: once for the antecedent and then again for the bound pronoun.[10]

Jäger (2001) proposes to deal with this situation by adding a new logical connective '|' and special (non-linear) inferences rules |E and |I (see page 105) for binding. Pronouns have category $np|np$; more generally, $X|Y$ means 'I function as something of category X, and I contain something of category Y that needs to be bound'. The general binding inference rule allows any expression of category Y to bind into the $X|Y$ as long as the binder precedes the bound expression.

One reason why Jäger's system is especially appropriate to consider here is that he explicitly recognizes the role of linear order in restricting binding, as we do, but unlike most LF-based accounts. Jäger's binding rule requires that the antecedent precede the pronoun it binds, and he gives empirical arguments that binding is sensitive to linear order. In particular, he correctly predicts weak crossover facts like those in (48). But he merely stipulates linear order: it follows from nothing, and the rule could just as easily have required that the binder follow its bound pronoun. Of course, Jäger is primarily interested in the role of contraction (duplication of resources), not linear order; but we would like to suggest that continuations can provide a deeper explanation for the role of linear order in crossover. On our account, the linear prohibition on weak crossover in quantificational binding follows from assuming that by default, people evaluate expressions from left to right.

In part to help compare our approach with Jäger's (2001), we also accomplish binding by means of a (single) non-linear inference rule. However, since (unlike Jäger) we are working within a multimodal TLG, rather than introducing a new logical connective |, we introduce a new modality $\circ_b$ for expressing binding relationships ('b' for 'binding'):

(49) $$\text{she} \vdash (np\backslash_b A)/\!\!/(np\backslash\!\!\backslash A).$$

[10]Dowty (1992) discusses binding and weak crossover in some early versions of TLG. In more recent but as yet unpublished work, he develops those ideas into an approach on which the duplication of resources is encapsulated entirely within the lexical meaning of the pronoun. As a result, the non-lexical part of the system, i.e., the logic proper, can remain linear. Like us, Dowty treats pronouns as quantificational, though he limits pronouns to taking scope only over types corresponding to verb phrases.

Conceptually, $np\backslash_b A$ is the category of an expression that would otherwise count as an A, but which contains a bindable pronoun. Thus the lexical entry for *she* turns any enclosing expression of category A into something of category $np\backslash_b A$. For instance, *John thought she left* would have category $np\backslash_b s$.

Our binding rule is a non-linear structural rule.[11]

(50) $$B \circledcirc K \vdash B \circ_b (B \circledcirc K) \quad \text{(Bind)}$$

It says that a subexpression B within a context $\mathcal{K}$ can bind into the larger expression $\mathcal{K}[B]$. We illustrate here by showing how to derive *Everyone$_i$ saw his$_i$ mother*, in which the subexpression *everyone* binds into *[everyone] saw his mother*.

(51)
$$\dfrac{\dfrac{\text{everyone} \vdash s/\!\!/(np\backslash\!\!\backslash s) \qquad \dfrac{\dfrac{\dfrac{\dfrac{np \vdash np}{}\,\text{Id} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{he} \vdash (np\backslash_b s)/\!\!/(np\backslash\!\!\backslash s) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots \atop np \circ (\text{saw} \circ (np \circ \text{'s mother})) \vdash s}{\Diamond(np \circ (\text{saw} \circ (np \circ \text{'s mother}))) \vdash \Diamond s}\,\Diamond\text{I} \qquad \dfrac{\dfrac{}{s \vdash s}\,\text{Id}}{\Diamond s \vdash s}\,\text{Unquote}}{\Diamond(np \circ (\text{saw} \circ (np \circ \text{'s mother}))) \vdash s}\,\Diamond\text{E}}{\Diamond np \circ \Diamond(\text{saw} \circ (np \circ \text{'s mother})) \vdash s}\,\text{K}'}{\Diamond np \circ (\Diamond\text{saw} \circ \Diamond(np \circ \text{'s mother})) \vdash s}\,\text{K}'}{\Diamond np \circ (\Diamond\text{saw} \circ (np \circ \text{'s mother})) \vdash s}\,\text{T}}{(\Diamond np \circ (\Diamond\text{saw} \circ (np \circ \text{'s mother}))) \circledcirc 1 \vdash s}\,\text{Push}}{(\Diamond\text{saw} \circ (np \circ \text{'s mother})) \circledcirc (1 \circ \Diamond np) \vdash s}\,\text{Right}}{(np \circ \text{'s mother}) \circledcirc ((1 \circ \Diamond np) \circ \Diamond\text{saw}) \vdash s}\,\text{Right}}{np \circledcirc (\text{'s mother} \circ ((1 \circ \Diamond np) \circ \Diamond\text{saw})) \vdash s}\,\text{Left}}{\text{'s mother} \circ ((1 \circ \Diamond np) \circ \Diamond\text{saw}) \vdash np\backslash\!\!\backslash s}\,\backslash\!\!\backslash\text{I}}{\text{he} \circledcirc (\text{'s mother} \circ ((1 \circ \Diamond np) \circ \Diamond\text{saw})) \vdash np\backslash_b s}\,/\!\!/\text{E}}{(\text{he} \circ \text{'s mother}) \circledcirc ((1 \circ \Diamond np) \circ \Diamond\text{saw}) \vdash np\backslash_b s}\,\text{Left}}{(\Diamond\text{saw} \circ (\text{he} \circ \text{'s mother})) \circledcirc (1 \circ \Diamond np) \vdash np\backslash_b s}\,\text{Right}}{(\Diamond np \circ (\Diamond\text{saw} \circ (\text{he} \circ \text{'s mother}))) \circledcirc 1 \vdash np\backslash_b s}\,\text{Right}}{(np \circ (\Diamond\text{saw} \circ (\text{he} \circ \text{'s mother}))) \circledcirc 1 \vdash np\backslash_b s}\,\text{T}}{(np \circ (\text{saw} \circ (\text{he} \circ \text{'s mother}))) \circledcirc 1 \vdash np\backslash_b s}\,\text{T}}{np \circledcirc ((\text{saw} \circ (\text{he} \circ \text{'s mother})) \circ 1) \vdash np\backslash_b s}\,\text{Left}}{np \circ_b (np \circledcirc ((\text{saw} \circ (\text{he} \circ \text{'s mother})) \circ 1)) \vdash s}\,\backslash_b\text{E}}{np \circledcirc ((\text{saw} \circ (\text{he} \circ \text{'s mother})) \circ 1) \vdash s}\,\text{Bind}}{(\text{saw} \circ (\text{he} \circ \text{'s mother})) \circ 1 \vdash np\backslash\!\!\backslash s}\,\backslash\!\!\backslash\text{I}}{\text{everyone} \circledcirc ((\text{saw} \circ (\text{he} \circ \text{'s mother})) \circ 1) \vdash s}\,/\!\!/\text{E}}{(\text{everyone} \circ (\text{saw} \circ (\text{he} \circ \text{'s mother}))) \circledcirc 1 \vdash s}\,\text{Left}}{\text{everyone} \circ (\text{saw} \circ (\text{he} \circ \text{'s mother})) \vdash s}\,\text{Pop}$$

A couple of points about this derivation: on this analysis, the pronoun itself is a scope-taking expression, taking scope over its binder's context. In this example, *everyone* binds *he*, and the context of *everyone* is *[ ] saw his mother*. (This context appears at the bottom of the derivation, where *everyone* enters, as $(\text{saw} \circ (\text{he} \circ \text{'s mother})) \circ 1$.) Thus *he* must take scope over this context—that is, must take as *its* context *np saw [ ]'s mother*, in which the

[11] If multimodal TLG supported not only multiplicative connectives but also exponentials in the linear-logic sense, we would replace the category *np* with the category $!_b np$ throughout our grammar (where $!_b$ is an exponential modality with respect to $\circ_b$; hence $!_b A \vdash A$ (dereliction) and $!_b A \vdash !_b A \circ_b !_b A$ (contraction)), and use the following binding rule, which is linear: $(B \circ_b C) \circledcirc K \vdash B \circ_b (C \circledcirc K)$.

type *np* plays the role of the placeholder variable bound by the quantifier *everyone*. (This context appears in the derivation, where *he* enters, as 's mother $\circ ((1 \circ \Diamond np) \circ \Diamond \text{saw})$.)

It is always easier to explain why a good derivation works than why some other sentence has no derivation, but let us offer a few words on why the crossover example **His*$_i$ *mother saw everyone*$_i$ (48b) does not have a bound reading. The crucial difference between (48a) and (48b) is that *everyone* is now to the right of the pronoun it is trying to bind. In order for *everyone* to take scope over the context *his mother saw [ ]*, we must apply the Right postulate twice (just as in the inverse-scope derivation above in (19)). But the Right postulate requires the introduction of at least two $\Diamond$s. Once diamonds enter the picture, the only way to remove them and unwrap the content of the quoted elements is by applying the Unquote postulate. But since Unquote only applies to expressions of type *s*, and there is no way to arrive at the type *s* without first unwrapping the pronoun, there is an unresolvable standoff: once the non-unquotable $np\backslash_b s$ gets quoted, the derivation is doomed.

7.2. **Questions and superiority.** As mentioned above, our analysis provides an explanation for superiority that parallels our explanation for weak crossover in relying on left-to-right evaluation order. Superiority is the name for the fact that a *wh*-trace must usually precede any additional, in-situ *wh*-phrase:

(52) a. Who saw what?
b. Who do you think __ saw what?
c. *What did who see __?

We suggest that in (52c), the presence of the additional *wh*-phrase *who* interferes with the ability of the fronted *wh*-phrase to bind its trace. The crucial assumption is that the *wh*-trace in (52c) can only get bound if it takes outermost scope over any additional *wh*-phrase. Once again, our basic strategy will be to show that the *wh*-trace can take scope over the in-situ *wh*-phrase only if it comes first.

Of course, before we can explain superiority, we have to first provide basic lexical entries for *wh*-words, and treat their usage both in-situ and raised. Just as we introduced a new modality $\circ_b$ above to express binding relationships, we now add a new modality $\circ_?$ to express questions:

(53) $$\text{who} \vdash (np\backslash_? S) /\!\!/ (np \backslash\!\!\backslash S),$$

(54) $$\text{what} \vdash (np\backslash_? S) /\!\!/ (np \backslash\!\!\backslash S),$$

(55) $$\text{which} \vdash ((np\backslash_? S) /\!\!/ (np \backslash\!\!\backslash S))/(np\backslash_n s).$$

Here the type variable S ranges over all types. Conceptually, $np\backslash_? S$ is the category of an S-question that seeks an *np*-answer. For example, a single-*wh* question has the type $np\backslash_? s$, and a double-*wh* question has the type $np\backslash_? np\backslash_? s$. The lexical entries above specify that *wh*-words take scope in situ, so they by themselves generate in-situ *wh*-questions. To generate raised-*wh* questions, we add two postulates.[12]

(56) $$A \circ_? (B \circ (C \circledcirc K)) \vdash A \circ_? (B \circledcirc (C \circ K)) \qquad \text{(Trace Left)}$$

(57) $$A \circ_? (C \circ (\Diamond B \circledcirc K)) \vdash A \circ_? (C \circledcirc (K \circ \Diamond B)) \qquad \text{(Trace Right)}$$

[12]These two postulates are really the composition of the Left and Right postulates with a single new postulate

$$A \circ_? (B \circ (\_\_ \circledcirc K)) \vdash A \circ_? (B \circledcirc K) \quad \text{(Trace)},$$

where __ is the empty string, in other words the identity for the default mode $\circ$. But as §5 explained, the default mode should not have an identity if we want *very* to have the type $(n/n)/(n/n)$. If multimodal TLG were to allow empty antecedents everywhere and use unary modalities to enforce non-emptiness when necessary, then we would be able to replace Trace Left and Trace Right with this Trace rule—an appealing prospect.

The following derivation shows the basic usage scenario for these additions to the grammar. On one hand, the entire derivation culminates in the pied-piped raised-*wh* question *Whose mother did Alice see?*. On the other hand, the $/\!\!/$E rule near the top concludes already with the in-situ *wh*-question *Alice saw whose mother?*.

(58)
$$
\begin{array}{cl}
\dfrac{\dfrac{}{np \vdash np}\,\text{Id} \qquad \dfrac{\text{who} \vdash (np\backslash_? s)/\!\!/(np\backslash\!\!\backslash s) \quad \begin{array}{c}\vdots\ \text{like the top of (51)}\\ \text{'s mother} \circ ((1 \circ \Diamond\text{Alice}) \circ \Diamond\text{saw}) \vdash np\backslash\!\!\backslash s\end{array}}{\text{who} \circledcirc (\text{'s mother} \circ ((1 \circ \Diamond\text{Alice}) \circ \Diamond\text{saw})) \vdash np\backslash_? s}\,/\!\!/\text{E}}{np \circ_? (\text{who} \circledcirc (\text{'s mother} \circ ((1 \circ \Diamond\text{Alice}) \circ \Diamond\text{saw}))) \vdash s}\,\backslash_?\text{E} & \\
\hline
np \circ_? ((\text{who} \circ \text{'s mother}) \circledcirc ((1 \circ \Diamond\text{Alice}) \circ \Diamond\text{saw})) \vdash s & \text{Left} \\
\hline
np \circ_? ((\text{who} \circ \text{'s mother}) \circ (\Diamond\text{saw} \circledcirc (1 \circ \Diamond\text{Alice}))) \vdash s & \text{Trace Right} \\
\hline
(\text{who} \circ \text{'s mother}) \circ (\Diamond\text{saw} \circledcirc (1 \circ \Diamond\text{Alice})) \vdash np\backslash_? s & \backslash_?\text{I} \\
\hline
(\text{who} \circ \text{'s mother}) \circ ((\Diamond\text{Alice} \circ \Diamond\text{saw}) \circledcirc 1) \vdash np\backslash_? s & \text{Right} \\
\hline
(\text{who} \circ \text{'s mother}) \circ (\Diamond\text{Alice} \circ \Diamond\text{saw}) \vdash np\backslash_? s & \text{Pop} \\
\hline
(\text{who} \circ \text{'s mother}) \circ (\text{Alice} \circ \Diamond\text{saw}) \vdash np\backslash_? s & \text{T} \\
\hline
(\text{who} \circ \text{'s mother}) \circ (\text{Alice} \circ \text{saw}) \vdash np\backslash_? s & \text{T}
\end{array}
$$

Of course, the result should be *Whose mother did Alice see?*, not *Whose mother Alice saw?*. We ignore subject-auxiliary inversion and verb forms purely for the sake of keeping the derivations simpler.

Let us now examine the derivation of a double-*wh* question that abides by superiority, so that we can see what goes wrong in an attempt to violate superiority. Below we derive the question *Who saw what?*, in which *who* is raised and *what* remains in situ. (We say that *who* is raised, even though it does not affect the string, because the derivation uses

Trace Left near the end. We use *Who saw what?* as our example for simplicity.)

(59)

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{(14)}
\UnaryInfC{$np \circ (\text{saw} \circ np) \vdash s$}
\RightLabel{$\Diamond$I}
\UnaryInfC{$\Diamond(np \circ (\text{saw} \circ np)) \vdash \Diamond s$}
\AxiomC{}
\RightLabel{Id}
\UnaryInfC{$s \vdash s$}
\RightLabel{Unquote}
\UnaryInfC{$\Diamond s \vdash s$}
\RightLabel{$\Diamond$E}
\BinaryInfC{$\Diamond(np \circ (\text{saw} \circ np)) \vdash s$}
\RightLabel{K$'$}
\UnaryInfC{$\Diamond np \circ \Diamond(\text{saw} \circ np) \vdash s$}
\RightLabel{K$'$}
\UnaryInfC{$\Diamond np \circ (\Diamond\text{saw} \circ \Diamond np) \vdash s$}
\RightLabel{T}
\UnaryInfC{$\Diamond np \circ (\Diamond\text{saw} \circ np) \vdash s$}
\RightLabel{Push}
\UnaryInfC{$(\Diamond np \circ (\Diamond\text{saw} \circ np)) \circledcirc 1 \vdash s$}
\RightLabel{Right}
\UnaryInfC{$(\Diamond\text{saw} \circ np) \circledcirc (1 \circ \Diamond np) \vdash s$}
\RightLabel{Right}
\UnaryInfC{$np \circledcirc ((1 \circ \Diamond np) \circ \Diamond\text{saw}) \vdash s$}
\RightLabel{$\backslash\!\!\backslash$I}
\UnaryInfC{$(1 \circ \Diamond np) \circ \Diamond\text{saw} \vdash np \backslash\!\!\backslash s$}
\AxiomC{$\text{what} \vdash (np \backslash_? s) /\!\!/ (np \backslash\!\!\backslash s)$}
\RightLabel{$/\!\!/$E}
\BinaryInfC{$\text{what} \circledcirc ((1 \circ \Diamond np) \circ \Diamond\text{saw}) \vdash np \backslash_? s$}
\RightLabel{Right}
\UnaryInfC{$(\Diamond\text{saw} \circ \text{what}) \circledcirc (1 \circ \Diamond np) \vdash np \backslash_? s$}
\RightLabel{Right}
\UnaryInfC{$(\Diamond np \circ (\Diamond\text{saw} \circ \text{what})) \circledcirc 1 \vdash np \backslash_? s$}
\RightLabel{Left}
\UnaryInfC{$\Diamond np \circledcirc ((\Diamond\text{saw} \circ \text{what}) \circ 1) \vdash np \backslash_? s$}
\RightLabel{T}
\UnaryInfC{$np \circledcirc ((\Diamond\text{saw} \circ \text{what}) \circ 1) \vdash np \backslash_? s$}
\RightLabel{T}
\UnaryInfC{$np \circledcirc ((\text{saw} \circ \text{what}) \circ 1) \vdash np \backslash_? s$}
\RightLabel{$\backslash\!\!\backslash$I}
\UnaryInfC{$(\text{saw} \circ \text{what}) \circ 1 \vdash np \backslash\!\!\backslash (np \backslash_? s)$}
\AxiomC{$\text{who} \vdash (np \backslash_? (np \backslash_? s)) /\!\!/ (np \backslash\!\!\backslash (np \backslash_? s))$}
\RightLabel{$/\!\!/$E}
\BinaryInfC{$\text{who} \circledcirc ((\text{saw} \circ \text{what}) \circ 1) \vdash np \backslash_? (np \backslash_? s)$}
\AxiomC{}
\RightLabel{Id}
\UnaryInfC{$np \vdash np$}
\RightLabel{$\backslash_?$E}
\BinaryInfC{$np \circ_? (\text{who} \circledcirc ((\text{saw} \circ \text{what}) \circ 1)) \vdash np \backslash_? s$}
\RightLabel{Trace Left}
\UnaryInfC{$np \circ_? (\text{who} \circ ((\text{saw} \circ \text{what}) \circledcirc 1)) \vdash np \backslash_? s$}
\RightLabel{$\backslash_?$I}
\UnaryInfC{$\text{who} \circ ((\text{saw} \circ \text{what}) \circledcirc 1) \vdash np \backslash_? (np \backslash_? s)$}
\RightLabel{Pop}
\UnaryInfC{$\text{who} \circ (\text{saw} \circ \text{what}) \vdash np \backslash_? (np \backslash_? s)$}
\end{prooftree}
$$

In the derivation above, Trace Left applies to the raised *wh*-phrase *who* in the context *[ ] saw what*. In other words, the raised *wh*-phrase takes scope over the rest of the sentence. Similarly, to derive the superiority violation **What did who see __?* (or **what who saw*, to ignore subject-auxiliary inversion), *what* must take scope over the context *who saw [ ]*. Just as in the previous section, taking scope over such a context requires using Right postulate twice, which introduces two ◇s and necessitates Unquote. But questions cannot be unquoted: their types are of the form $A\backslash_? S$, not $\Diamond_u S$. Superiority is thus enforced.

For Jäger, *wh*-phrases have category $q/(np \uparrow s)$, where q is the category of questions and "$np \uparrow s$" is an s with an np-gap. Because connecting gaps with their fillers uses a mechanism separate from binding, whatever might explain superiority in Jäger's system will be independent of the explanation for weak crossover. On our account, connecting a *wh*-trace with its filler uses the same continuation-based mechanism as binding, and the explanation for why an intervening in-situ *wh*-word prevents a fronted *wh*-phrase from binding its trace follows from the same evaluation order mechanism as weak crossover.

Whether or not weak crossover and superiority ought to have a unified explanation is debatable, and further investigation may reveal empirical arguments that these two phenomena are in fact distinct. We will not attempt to settle that question here; our main point is that we cannot even have the debate in TLG unless continuations are made available.

As for empirical coverage, there is an interesting interaction between *wh*-binding and weak crossover that we call *pied binding*, which seems to favor our unified treatment.

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

To a first approximation, a *wh*-phrase can bind a pronoun only if the trace of the *wh*-phrase precedes the pronoun. One explanation that has been popular at least since Reinhart (1983) is that the trace itself does the binding. On Jäger's system, the trace can indeed accomplish the binding in examples such as (60a) (see especially pages 124–125). The unacceptability of (60b) is also explained, since when the trace follows the pronoun, the prohibition against cataphora predicts the relative unacceptability of (60b).

But there are more complex examples which cast doubt on the assumption that it is the trace that is doing the binding:

(61) a. Whose$_i$ father did John say __ saw his$_i$ mother?
b. *Whose$_i$ father did John say his$_i$ mother saw __?

In each case, the trace is bound by the entire pied-piped *wh*-phrase *whose father*. That is, in each case, it is a father who is seeing or being seen. Yet the pronoun can be bound by the *wh*-word alone, as indicated by the indexation in (61a), suggesting that the *wh*-word must be able to bind the pronoun directly after all.

Interestingly, (61b) shows that binding still requires linear precedence of a sort: the pronoun can be bound by the *wh*-word *whose* only if the *wh*-trace precedes the pronoun. In our system, *whose* can bind a pronoun just as a quantificational NP does. But just as superiority is violated when an in-situ *wh*-phrase interferes between a raised *wh*-phrase and its trace, weak crossover is perpetrated when a bindable pronoun interferes between a raised *wh*-phrase and its trace. Put differently, (61b) constitutes a weak crossover violation because the corresponding in-situ *wh*-question *John said his$_i$ mother saw whose$_i$ father?* does. This complex pattern of interactions between pied-piping, binding of *wh*-traces, and binding of pronouns falls out from our unified account without additional stipulation.

This brief discussion hardly counts as a thorough comparison of Shan and Barker 2003 with Jäger 2001; that would take far too much space here. Our only intention is to suggest that a continuation-based analysis of a complex phenomenon can hold its own against a robust previous TLG account, and to point out that our account provides a different pattern of explanation. In particular, we connect the linear order asymmetry of weak crossover with the computational concept of evaluation order; and we also provide an analysis on which weak crossover and superiority have distinct but closely parallel explanations. Thus continuations constitute a new and potentially valuable explanatory tool for TLG analyses.

7.3. **Polarity licensing.** Polarity sensitivity (Ladusaw 1979) has been a popular linguistic phenomenon to analyze in the type-logical (Bernardi 2002; Bernardi and Moot 2001), categorial (Dowty 1994), and lexical-functional (Fry 1997, 1999) approaches to grammar. The multitude of these analyses is in part due to the more explicit emphasis that these traditions place on the syntax-semantics interface—be it in the form of the Curry-Howard isomorphism, Montague-style semantic rules, or linear logic as glue—and the fact that polarity sensitivity is a phenomenon that spans syntax and semantics.

On one hand, which polarity items are licensed or prohibited in a given linguistic environment depends, by and large, on semantic properties of that environment (Krifka 1995; Ladusaw 1979, inter alia). For example, to a first approximation, negative polarity items can occur only in *downward-entailing* contexts, such as under the scope of a *monotonically decreasing* quantifier. A quantifier q, of type $(np \to s) \to s$, is monotonically decreasing just in case

(62) $$\forall s_1.\, \forall s_2.\, (\forall x.\, s_2(x) \Rightarrow s_1(x)) \Rightarrow q(s_1) \Rightarrow q(s_2).$$

Thus (63a) is acceptable because the scope of *no one* is downward-entailing, whereas (63b) and (63c) are unacceptable.

(63) a. No one saw anyone.
b. *Everyone saw anyone.
c. *Alice saw anyone.

To account for these contrasts, Fry (1997, 1999), Bernardi (2002), and Bernardi and Moot (2001) all split the clause type s into several types, related by subtyping. Fry (1997, 1999) distinguishes between s and $\ell \multimap (s \otimes \ell)$ in linear logic, where ℓ stands for polarity licensing as a grammatical resource. Analogously, Bernardi (2002) and Bernardi and Moot (2001) distinguish between different unary modalities applied to s.

A simplistic version of Bernardi's analysis is to distinguish between the clause types s ('neutral clause') and $\Box_p^{\downarrow}\Diamond_p s$ ('negative clause'). Here $\Diamond_p$ is a new unary mode (the letter p stands for "polarity"), and we abbreviate $\Box_p^{\downarrow}\Diamond_p s$ to s^-. We choose the type $\Box_p^{\downarrow}\Diamond_p s$ so that $s \vdash s^-$ is a theorem in multimodal TLG.

(64) $$\dfrac{\dfrac{}{\Diamond_p s \vdash \Diamond_p s}\,\mathrm{Id}}{s \vdash \Box_p^{\downarrow}\Diamond_p s}\,\Box_p^{\downarrow}\mathrm{I}$$

Splitting the clause type like this helps us account for polarity sensitivity, because we can now assign different types to different quantifiers in the lexicon:

(65) everyone $\vdash s \mathbin{/\!\!/} (np \mathbin{\backslash\!\!\backslash} s)$, no one $\vdash s \mathbin{/\!\!/} (np \mathbin{\backslash\!\!\backslash} s^-)$, anyone $\vdash s^- \mathbin{/\!\!/} (np \mathbin{\backslash\!\!\backslash} s^-)$.

The type of *everyone* is unchanged: it takes scope over a neutral clause to form a neutral clause. The types of *no one* and *anyone* involve the newly introduced s^-: they both take scope over a negative clause, but *no one* forms a neutral clause whereas *anyone* forms a negative clause. As (64) shows, a neutral clause can be converted to a negative clause. Thus, for example, *anyone* can take scope in $np \circ (\mathrm{saw} \circ \mathrm{anyone})$ to form a negative clause s^-, even though the verb *saw* produces a neutral clause s initially.

(66) $$\dfrac{\dfrac{\dfrac{\mathrm{anyone} \vdash s^- \mathbin{/\!\!/} (np \mathbin{\backslash\!\!\backslash} s^-) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{np \circ (\mathrm{saw} \circ np) \vdash s}\,(14)}{\Diamond_p(np \circ (\mathrm{saw} \circ np)) \vdash \Diamond_p s}\,\Diamond_p\mathrm{I}}{np \circ (\mathrm{saw} \circ np) \vdash s^-}\,\Box_p^{\downarrow}\mathrm{I}}{(np \circ (\mathrm{saw} \circ np)) \circledcirc 1 \vdash s^-}\,\mathrm{Push}}{(\mathrm{saw} \circ np) \circledcirc (1 \circ np) \vdash s^-}\,\mathrm{Right}}{np \circledcirc ((1 \circ np) \circ \mathrm{saw}) \vdash s^-}\,\mathrm{Right}}{(1 \circ np) \circ \mathrm{saw} \vdash np \mathbin{\backslash\!\!\backslash} s^-}\,\mathbin{\backslash\!\!\backslash}\mathrm{I}}{\mathrm{anyone} \circledcirc ((1 \circ np) \circ \mathrm{saw}) \vdash s^-}\,\mathbin{/\!\!/}\mathrm{E}}{(\mathrm{saw} \circ \mathrm{anyone}) \circledcirc (1 \circ np) \vdash s^-}\,\mathrm{Right}}{\dfrac{(np \circ (\mathrm{saw} \circ \mathrm{anyone})) \circledcirc 1 \vdash s^-}{np \circ (\mathrm{saw} \circ \mathrm{anyone}) \vdash s^-}\,\mathrm{Pop}}\,\mathrm{Right}$$

As before, we consider a derivation complete if it culminates in the type s, not s^-. The fact that $np \circ (\mathrm{saw} \circ \mathrm{anyone})$ only has the type s^- (as derived above) and not s predicts that a clause like **Alice saw anyone* (63c) is not a grammatical sentence by itself. Moreover, because there is no way to convert a negative clause to a neutral clause, *everyone* cannot take scope over a negative clause, so **Everyone saw anyone* (63b) is ruled out as well. However, *no one* (unlike *Alice* or *everyone*) can take scope over *anyone* to form a complete (neutral) clause. Thus even our simplistic rendering of Bernardi's account predicts correctly that *No one saw anyone* (63a) has a linear-scope reading (but no inverse-scope reading). We also predict correctly that the sentence

(67) No one introduced everyone to anyone.

has no linear-scope reading, even though it is acceptable when *no one* scopes over *anyone*, then *everyone*. (Szabolcsi (2004) discusses such "intervention" cases.)

On the other hand, a restriction on surface syntactic form, such as that imposed by polarity sensitivity, is by definition a matter of syntax. Besides, there are syntactic restrictions on the configuration relating the licensor to the licensee. For example, (63a) above is acceptable—*no one* manages to license *anyone*—but (68) below is not. As the contrasts in (69) and (70) further illustrate, the licensor usually needs to precede the licensee.

(68) *Anyone saw no one.

(69) a. Nobody's mother saw anybody's father.
b. *Anybody's mother saw nobody's father.

(70) a. Alice didn't visit anyone.
b. *Anyone didn't visit Alice.

These and other examples show that the syntactic relations allowed between licensor and licensee for polarity purposes are similar to those allowed between antecedent and pronoun for binding purposes. Because Fry, Bernardi, and Bernardi and Moot focus on quantification and scope, they easily characterize how *no* must scope over *any* in order to license it, but they leave it a mystery why *no* must precede *any* in order to license it. In particular, they wrongly accept all of (68–70). This mystery has been noted by Ladusaw (1979, §9.2), and Fry (1999, §8.2) likewise identifies it as a defect in current accounts of polarity sensitivity.

Continuations provide precisely the missing link between linear order and quantifier scope. Recall from §6.3 that inverse scope can be generated, even under a regime of left-to-right evaluation, using the Unquote rule in (46). The crucial step, as illustrated in (47), is to unquote the clause produced by the narrower-scope quantifier. In that example, *everyone* can take inverse scope over *someone*, because *someone* produces a neutral clause s. A neutral clause can be unquoted because its type s is shorthand for $\Diamond_u s'$, which is enclosed in $\Diamond_u$. By contrast, a negative clause cannot be unquoted because its type s^- is shorthand for $\Box_p^{\downarrow}\Diamond_p\Diamond_u s'$, which is not enclosed in $\Diamond_u$. Given that *anyone* produces s^-, then, inverse scope over *anyone* is impossible. This prediction correctly rules out the unacceptable examples in (68–70) while leaving (67) available.

This explanation for linear-order effects in polarity licensing is further developed elsewhere (Shan 2004). We are not aware of any other account of polarity sensitivity that unifies its syntactic properties with weak crossover as we do using left-to-right evaluation.

7.4. **Reversing evaluation order.** One way to see that our implementation of evaluation order unifies linear order effects in weak crossover, superiority, and polarity is to impose right-to-left evaluation and see what happens. Because our Left and Right postulates embody left-to-right evaluation order, we can reverse default evaluation order while leaving all other aspects of the system unchanged.

More specifically, suppose that we replace the Left and Right postulates with

(71) $B \circledcirc (\Diamond C \circ K) \dashv\vdash (B \circ \Diamond C) \circledcirc K$ (Left), $(B \circ C) \circledcirc K \dashv\vdash C \circledcirc (K \circ B)$ (Right).

We can still derive linear and inverse scope for quantifiers, but the predictions concerning the weak crossover judgments in (48), the superiority judgments in (52), and the polarity judgments in (68–70) reverse: binders must follow any pronouns they bind, *wh*-traces must follow any additional *wh*-phrases, and polarity triggers must follow the polarity items they license.

8. Conclusions

Let's take stock. Our core proposal implements the full power of the quantificational type constructor q using only a single new mode ($\circledcirc$) along with two structural postulates, Left and Right. (If empty antecedents are forbidden, we also need to simulate Push and Pop.) We provided a geometric interpretation of types to explain how this parsimonious reduction of q constitutes an implementation of continuations: under the geometric interpretation, the continuation mode decomposes types into a subexpression in the foreground and a remainder in the background, where the remainder is interpreted as a context.

We then (§6.3) imposed left-to-right evaluation order by stipulating that a quantificational element could only take scope over an expression to its left if that expression had no side effects, i.e., was "pure", as indicated by the presence of the unary modality $\diamond$. In order to restore inverse scope, we added an Unquote rule to remove the $\diamond$, though only for expressions in category s. Under the analogy with computation, the evaluation of an expression is delayed for as long as it is under the quotation modality $\diamond$. By quoting a leftmost quantifier and then unquoting it later, we can delay the evaluation of the quantifier and let it take narrow (hence inverse) scope with respect to some following quantifier.

We proposed analyses on which pronominal binding (§7.1), the binding of a *wh*-trace (§7.2), and licensing of a negative polarity item (§7.3) were all accomplished via side-effects. Because bindable pronouns have their own side-effects, their presence disrupted the unquotation strategy, so that the analysis automatically predicts that a quantifier can take scope over a pronoun to its left but cannot bind it. That is, the analysis correctly allows inverse scope but not weak crossover violations. Similarly, the analysis predicts that a *wh*-phrase can bind its trace only when no other *wh*-phrase intervenes. That is, the analysis correctly allows multiple *wh*-questions but not superiority violations. Finally, the analysis automatically correctly requires a negative polarity licensor to precede its licensee.

We believe that the theoretical parsimony and empirical robustness of these analyses show how continuations offer new and valuable tools for multimodal type-logical grammar writers. In particular, continuation-based analyses offer new insights into quantification, binding, multiple *wh*-questions, and negative polarity.

References

Aoun, Joseph, and Yen-hui Audrey Li. 1993. *Syntax of scope*. Cambridge: MIT Press.

Barendregt, Henk P. 1981. *The lambda calculus: Its syntax and semantics*. Amsterdam: Elsevier Science.

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

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

———. 2004. Continuations in natural language (extended abstract). In Thielecke (2004), 1–11.

Bernardi, Raffaella. 2002. Reasoning with polarity in categorial type logic. Ph.D. thesis, Utrecht Institute of Linguistics (OTS), Utrecht University.

———. 2003. CTL: In situ binding. http://www.let.uu.nl/~Raffaella.Bernardi/personal/q_solutions.pdf.

Bernardi, Raffaella, and Richard Moot. 2001. Generalized quantifiers in declarative and interrogative sentences. *Journal of Language and Computation* 1(3):1–19.

Dalrymple, Mary, ed. 1999. *Semantics and syntax in Lexical Functional Grammar: The resource logic approach*. Cambridge: MIT Press.

Dalrymple, Mary, John Lamping, Fernando Pereira, and Vijay A. Saraswat. 1999. Quantification, anaphora, and intensionality. In Dalrymple (1999), chap. 2, 39–89.

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

Dowty, David. 1992. 'Variable-free' syntax, variable-binding syntax, the natural deduction Lambek calculus, and the crossover constraint. In *Proceedings of the 11th West Coast Conference on Formal Linguistics*, ed. Jonathan Mead. Stanford, CA: Center for the Study of Language and Information.

Dowty, David R. 1994. The role of negative polarity and concord marking in natural language reasoning. In *SALT IV: Semantics and linguistic theory*, ed. Mandy Harvey and Lynn Santelmann. Ithaca: Cornell University Press.

Felleisen, Matthias. 1988. The theory and practice of first-class prompts. In *POPL '88: Conference record of the annual ACM symposium on principles of programming languages*, 180–190. New York: ACM Press.

Fry, John. 1997. Negative polarity licensing at the syntax-semantics interface. In *Proceedings of the 35th annual meeting of the Association for Computational Linguistics and 8th conference of the European chapter of the Association for Computational Linguistics*, ed. Philip R. Cohen and Wolfgang Wahlster, 144–150. San Francisco: Morgan Kaufmann.

———. 1999. Proof nets and negative polarity licensing. In Dalrymple (1999), chap. 3, 91–116.

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

Huang, Cheng-Teh James. 1982. Logical relations in Chinese and the theory of grammar. Ph.D. thesis, Department of Linguistics and Philosophy, Massachusetts Institute of Technology.

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

Jäger, Gerhard. 2001. Anaphora and type logical grammar. Habilitationsschrift, Humboldt University Berlin. UIL-OTS Working Papers 01004-CL/TL, Utrecht Institute of Linguistics (OTS), Utrecht University.

Krifka, Manfred. 1995. The semantics and pragmatics of polarity items. *Linguistic Analysis* 25:209–257.

Ladusaw, William A. 1979. Polarity sensitivity as inherent scope relations. Ph.D. thesis, Department of Linguistics, University of Massachusetts. Reprinted by New York: Garland, 1980.

Lambek, Joachim. 1958. The mathematics of sentence structure. *American Mathematical Monthly* 65(3):154–170.

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

Computer Science 193, Berlin: Springer-Verlag.

Moortgat, Michael. 1988. *Categorial investigations: Logical and linguistic aspects of the Lambek calculus*. Dordrecht: Foris.

———. 1995. In situ binding: A modal analysis. In *Proceedings of the 10th Amsterdam Colloquium*, ed. Paul Dekker and Martin Stokhof, 539–549. Institute for Logic, Language and Computation, Universiteit van Amsterdam.

———. 1996a. Categorial type logics. In *Handbook of logic and language*, ed. Johan van Benthem and Alice ter Meulen, chap. 2. Amsterdam: Elsevier Science.

———. 1996b. Generalized quantification and discontinuous type constructors. In *Discontinuous constituency*, ed. Harry C. Bunt and Arthur van Horck, 181–207. Berlin: Mouton de Gruyter.

———. 2000. Computational semantics: Lab sessions. `http://www.let.uu.nl/~Michael.Moortgat/personal/Courses/cslab2000s.pdf`.

Moot, Richard. 2002. Proof nets for linguistic analysis. Ph.D. thesis, Utrecht Institute of Linguistics (OTS), Utrecht University.

Morrill, Glyn V. 1994. *Type logical grammar: Categorial logic of signs*. Dordrecht: Kluwer.

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

Parigot, Michel. 1992. $\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction. In *Proceedings of LPAR '92: International conference on logic programming and automated reasoning*, ed. Andrei Voronkov, 190–201. Lecture Notes in Artificial Intelligence 624, Berlin: Springer-Verlag.

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

Restall, Greg. 2000. *An introduction to substructural logics*. London: Routledge.

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

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

Shan, Chung-chieh, and Chris Barker. 2003. Explaining crossover and superiority as left-to-right evaluation. Draft manuscript, Harvard University and University of California, San Diego; `http://semanticsarchive.net/Archive/TBjZDQ3Z/`.

Sitaram, Dorai, and Matthias Felleisen. 1990. Control delimiters and their hierarchies. *Lisp and Symbolic Computation* 3(1):67–99.

Steedman, Mark J. 2000. *The syntactic process*. Cambridge: MIT Press.

Szabolcsi, Anna. 1997. Strategies for scope taking. In *Ways of scope taking*, ed. Anna Szabolcsi, chap. 4, 109–154. Dordrecht: Kluwer.

———. 2004. Positive polarity—negative polarity. *Natural Language and Linguistic Theory* 22(2):409–452.

Taha, Walid, and Michael Florentin Nielsen. 2003. Environment classifiers. In *POPL '03: Conference record of the annual ACM symposium on principles of programming languages*, 26–37. New York: ACM Press.

Thielecke, Hayo, ed. 2004. *CW'04: Proceedings of the 4th ACM SIGPLAN workshop on continuations*. Tech. Rep. CSR-04-1, School of Computer Science, University of Birmingham.

Department of Linguistics, University of California, San Diego
E-mail address: barker@ucsd.edu
URL: http://www.ling.ucsd.edu/~barker/

Division of Engineering and Applied Sciences, Harvard University
E-mail address: ccshan@post.harvard.edu
URL: http://www.digitas.harvard.edu/~ken/