Continuations and Natural Language

OXFORD STUDIES IN THEORETICAL LINGUISTICS

Recent Titles

34 Dissolving Binding Theory
by Johan Rooryck and Guido Vanden Wyngaerd

35 The Logic of Pronominal Resumption
by Ash Asudeh

36 Modals and Conditionals
by Angelika Kratzer

37 The Theta System
Argument Structure at the Interface
edited by Martin Everaert, Marijana Marelj, and Tal Siloni

38 Sluicing
Cross-Linguistic Perspectives
edited by Jason Merchant and Andrew Simpson

39 Telicity, Change, and State
A Cross-Categorial View of Event Structure
edited by Violeta Demonte and Louise McNally

40 Ways of Structure Building
edited by Myriam Uribe-Etxebarria and Vidal Valmala

41 The Morphology and Phonology of Exponence
edited by Jochen Trommer

42 Count and Mass Across Languages
edited by Diane Massam

43 Genericity
edited by Alda Mari, Claire Beyssade, and Fabio Del Prete

44 Strategies of Quantification
edited by Kook-Hee Gil, Steve Harlow, and George Tsoulas

45 Nonverbal Predication
Copular Sentences at the Syntax-Semantics Interface
by Isabelle Roy

46 Diagnosing Syntax
edited by Lisa Lai-Shen Cheng and Norbert Corver

47 Pseudogapping and Ellipsis
by Kirsten Gengel

48 Syntax and its Limits
edited by Raffaella Folli, Christina Sevdali, and Robert Truswell

49 Phrase Structure and Argument Structure
A Case Study of the Syntax-Semantics Interface
by Terje Lohndal

50 Edges in Syntax
Scrambling and Cyclic Linearization
by Heejeong Ko

51 The Syntax of Roots and the Roots of Syntax
edited by Artemis Alexiadou, Hagit Borer, and Florian Schäfer

52 Causation in Grammatical Structures
edited by Bridget Copley and Fabienne Martin

53 Continuations and Natural Language
by Chris Barker and Chung-chieh Shan

For a complete list of titles published and in preparation for the series, see pp. ~~xxx–x~~.

Continuations and Natural Language

CHRIS BARKER AND CHUNG-CHIEH SHAN

OXFORD
UNIVERSITY PRESS

OXFORD
UNIVERSITY PRESS

Great Clarendon Street, Oxford OX2 6DP
United Kingdom

Oxford University Press is a department of the University of Oxford.
It furthers the University's objective of excellence in research, scholarship,
and education by publishing worldwide. Oxford is a registered trade mark of
Oxford University Press in the UK and in certain other countries

First Edition published in 2014

Impression: 1

Published in the United States of America by Oxford University Press
198 Madison Avenue, New York, NY 10016, United States of America

British Library Cataloguing in Publication Data

Data available

Library of Congress Control Number: 2014937427

ISBN 978–0–19–957501–5 (hbk.)
978–0–19–957502–2 (pbk.)

Printed and bound by
CPI Group (UK) Ltd, Croydon CR0 4YY

Contents

General preface

The theoretical focus of this series is on the interfaces between subcomponents of the human grammatical system and the closely related area of the interfaces between the different subdisciplines of linguistics. The notion of 'interface' has become central in grammatical theory (for instance, in Chomsky's Minimalist Program) and in linguistic practice: work on the interfaces between syntax and semantics, syntax and morphology, phonology and phonetics, etc. has led to a deeper understanding of particular linguistic phenomena and of the architecture of the linguistic component of the mind/brain.

The series covers interfaces between core components of grammar, including syntax/morphology, syntax/semantics, syntax/phonology, syntax/pragmatics, morphology/phonology, phonology/phonetics, phonetics/speech processing, semantics/pragmatics, and intonation/discourse structure, as well as issues in the way that the systems of grammar involving these interface areas are acquired and deployed in use (including language acquisition, language dysfunction, and language processing). It demonstrates, we hope, that proper understandings of particular linguistic phenomena, languages, language groups, or inter-language variations all require reference to interfaces.

The series is open to work by linguists of all theoretical persuasions and schools of thought. A main requirement is that authors should write so as to be understood by colleagues in related subfields of linguistics and by scholars in cognate disciplines.

The interactions between linear order, syntactic structure, and semantic scope have been at the heart of generative grammar for decades. In this monograph, Barker and Shan delineate a new perspective on this problem that appeals to the notion that the semantic context of a quantificational expression can, in effect, function as an argument of that expression, and that the natural language phenomena that have been so puzzling over the years can be understood as deriving from the order of semantic evaluation of such contexts. This perspective allows the authors to weave together the puzzling structural and linear ordering effects that have been discovered over the years into a unified theoretical explanation, which they extend to related phenomena in natural language.

David Adger
Hagit Borer

Notational conventions

$\mathtt{e} \to \mathtt{t}$ Types: Instead of writing $\langle \tau, \sigma \rangle$ for the type of a function from objects of type τ into objects of type σ, we follow the convention in computer science, and write $\tau \to \sigma$. So the type of an extensional verb phrase will be $\mathtt{e} \to \mathtt{t}$.

saw j m Associativity for values: Values associate from left-to-right. An expression written **saw j m** is equivalent to (**saw j**) **m**.

$\mathtt{e} \to \mathtt{e} \to \mathtt{t}$ Associativity for types: Since values associate from left-to-right, types correspondingly associate from right-to-left. This means that the type $\mathtt{e} \to \mathtt{e} \to \mathtt{t}$ is shorthand for $\mathtt{e} \to (\mathtt{e} \to \mathtt{t})$ (an extensional transitive verb). The type of an extensional generalized quantifier must be written with parentheses, i.e., $(\mathtt{e} \to \mathtt{t}) \to \mathtt{t}$.

$\lambda abc.M$ Dot notation for lambda abstraction: We adopt the standard convention that $\lambda a \lambda b \lambda c M$ can be abbreviated as $\lambda abc.M$. We will assume that the scope of the lambdas before the dot extends as far to the right as possible. Therefore the expression $\lambda x.$ **yesterday** (**call** x) **m** is equivalent to $\lambda x\big((\textbf{yesterday}\ (\textbf{call}\ x))\ \textbf{m}\big)$.

$g[\ ]$ Contexts, holes, and plugs: Throughout the book, we will consider logical expressions that have a hole ('[ ]') somewhere in them, e.g., $\lambda x.(x[\ ])y$. We will say that an expression that contains exactly one hole is a (certain kind of) *context*. Contexts can have their holes plugged. This means replacing the hole ('[ ]') with some expression. For instance, if $g[\ ] = \lambda x.(x[\ ])y$, then $g[w] = \lambda x.(xw)y$. Plugs can be complex expressions ($g[(ww)] = \lambda x.(x(ww))y$). We allow a context to be plugged by a context. For instance, if we plug the hole in $g[\ ]$ with itself, we get $g[g[\ ]] = \lambda x.x(\lambda x.(x[\ ])y)y$, which has exactly one hole in it, so the net result is once again a context. See sections 1.3 and 13.2, as well as Shan (2005):26 ff., and Shan (2005) chapter 3 for additional discussion.

Acknowledgments

With Barker (2001, 2002) and Shan (2001a,b) as starting points, Part I is largely based on joint work reported in Shan and Barker (2006), Barker and Shan (2006), and Barker and Shan (2008), as developed and elaborated in Shan (2004, 2005) and in Barker (2009, 2012, 2014a). Part II is based on Barker (2007) and Barker (2013). The technical results in chapter 17 have not previously appeared, and constitute new joint work. With the exception of that chapter, most of the writing and preparation of this book was (joyfully) undertaken by the first author (Barker), relying on the inspiration, advice, and commentary of the second author (Shan).

Most of the material in this book has benefited from feedback from courses, tutorials, conferences, and workshops. Special thanks to Sascha Bargmann, Raffaella Bernardi, Dylan Bumford, Lucas Champollion, Simon Charlow, John Davey, Paul Elbourne, Svetlana Godjevac, Christopher Götze, Philippe de Groote, Daniel Gutzmann, Polly Jacobson, Chris Kennedy, Oleg Kiselyov, Dan Lassiter, Tim Leffel, Salvador Mascarenhas, Michael Moortgat, Jim Pryor, Mike Solomon, Anna Szabolcsi, Michael Tabatowski, Dylan Thurston, Gert Webelhuth, ESSLLIs in 2004, 2005, and 2007, and colleagues at Goethe University who field-tested a draft of this book.

We're grateful to Anna Szabolcsi for urging us to make the system easier to understand, pressure which more or less directly resulted in creating the tower notation; and to Oleg Kiselyov, who suggested using contexts to simplify the semantic part of the tower notation.

List of abbreviations

AR	Argument Raising
CBN	call-by-name Continuation Passing style
CBV	call-by-value Continuation Passing style
CL	Combinatory Logic
CPS	Continuation Passing style
DMG	Dynamic Montague Grammar
DP	determiner phrase
DPL	Dynamic Predicate Logic (Groenendijk and Stokhof 1991)
I	identity function (λxx)
κ	variable over continuations
LF	Logical Form
LG	Lambek-Grishin logic (Bernardi and Moortgat 2010)
N	noun
NEG	negation
NL_{CL}	non-associative Lambek grammar (NL) augmented with Combinatory Logic
NL_{λ}	non-associative Lambek grammar (NL) augmented with λ-abstraction
NP	noun phrase
NPI	negative polarity item
$\mathcal{P}$	variable over pronoun meanings (category pn)
PF	Phonetic Form
pn	category of a simple pronoun $\dfrac{\mathrm{DP} \triangleright \mathrm{S} \,\vert\, \mathrm{S}}{\mathrm{DP}}$
PP	prepositional phrase
PTQ	'proper treatment of quantification' (Montague 1974)
Q	DP?S (the kind of question that asks for an individual)
QP	quantificational DP
QR	Quantifier Raising
S	clause/sentence
S^{-}	a proper subset of the category S
VP	verb phrase
VPE	verb phrase ellipsis
VR	Value Raising

Introduction

This book is about continuations. It argues that continuations are an essential component of a complete understanding of natural language meaning.

(1) **The continuation hypothesis**: some natural language expressions denote functions on their continuations, i.e., functions that take their own semantic context as an argument.

The main way that we will argue in favor of this hypothesis is by providing analyses of a variety of natural language phenomena whose insights depend on explicit reference to continuations.

What is a continuation?

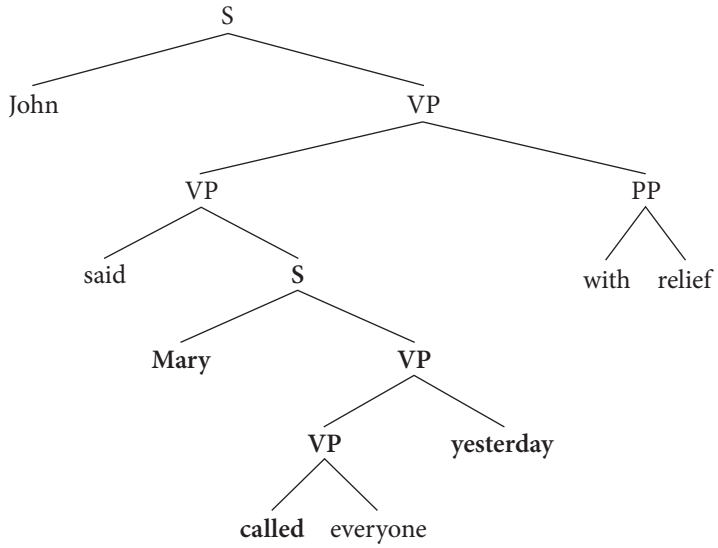
A **continuation** is a portion of the context surrounding an expression.

(2) John said [**Mary called** everyone **yesterday**] with relief.

In (2), the continuation of the expression *everyone*, relative to the bracketed embedded clause that contains it, is the remainder of the embedded clause after *everyone* has been removed. This material includes the lexical items *Mary*, *called*, and *yesterday*.

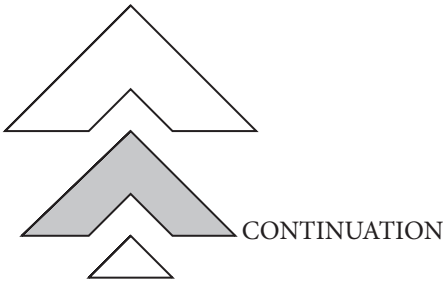
Based on the string presentation, this might appear to be a discontinuous object, but the contiguous nature of the continuation becomes immediately apparent when we consider the syntactic phrase structure tree:

(3)



In this tree, the continuation of *everyone* relative to the embedded clause is the contiguous portion of the tree dominating *Mary*, *called*, and *yesterday*, but with *everyone* removed. More schematically, we have the following diagram:

(4)



In the example at hand, the unshaded upper notched triangle corresponds to the portion of the structure in which the smaller clause is embedded, and includes *John*, *said*, and *with relief*. The middle gray notched triangle corresponds to the material over which the scope-taker *everyone* takes scope—its continuation. And the smallest unshaded unnotched triangle corresponds to the scope-taker *everyone*.

We will use schematic diagrams like this one from time to time throughout the book. We'll call them *tangram diagrams*, after the puzzle game in which a set of flat geometric shapes are rearranged into a variety of larger shapes.

Since we will be primarily concerned with meaning, we will concentrate on *semantic* context, rather than, say, phonological context or syntactic context. In the example above, then, the semantic continuation of *everyone* relative to the bracketed clause is the meaning of that clause with the contribution of *everyone* abstracted, namely, the property of being called yesterday by Mary, which can be rendered as $\lambda x.\mathbf{yesterday}(\mathbf{called}\ x)\ \mathbf{m}$.

What makes continuations essential?

The semantic continuation identified in (2) is what the quantifier *everyone* takes for its semantic argument, its 'nuclear scope'. In general, identifying the semantic argument of a scope-taking expression is the same thing as identifying (one of its) continuations. Thus scope-taking is the most compelling application of continuations in natural language.

But it is not enough for us to show that continuations provide an elegant way to conceptualize scope-taking, given that there are other effective strategies for handling scope-taking that do not explicitly involve continuations, such as Quantifier Raising, Flexible Montague Grammar, and so on. To make a strong case that continuations are essential, we must argue that continuations provide insights that are not available in other approaches.

We find inspiration for such insights in neighboring disciplines. Continuations are an idea that has been explored in some depth in the theory of computer programming languages, where they have been used (among many applications) to characterize **order of evaluation** of expressions in a computer program, as explained in section 12.1. And in general, one of the distinctive advantages of continuations is that they provide a way to reason explicitly about the order in which a computation unfolds.

We will argue that a number of phenomena in natural language depend on order of evaluation. These include quantificational binding, crossover, reconstruction, negative polarity licensing, and donkey anaphora.

In particular, one of the central results will be a robust explanation of crossover in terms of order of evaluation.

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

When the quantifier *everyone* precedes the pronoun *his*, as in (5a), the binding relationship indicated by the subscripts is available, in which case (5a) expresses a generalization about filial duty: every person loves that person's mother. But when the quantifier follows the pronoun, as in (5b), the indicated binding relationship is not possible: (5b) cannot express a general thought about family relationships, namely, that each person's mother loves that person. We will say that (5b) is a weak crossover violation.

Ever since Reinhart (1983), standard approaches to crossover (e.g., Büring 2005) have rejected the relevance of linear order in favor of purely hierarchical relationships based on c-command. However, as explained in chapter 2, following Shan and Barker (2006) and Barker (2012), we believe that c-command is not a requirement for quantificational binding. As a result, we endorse a minority tradition including Bresnan (1994, 1998), Safir (2004a,b), and Jäger (2005), who argue that in English and in many other languages, some kind of order plays a role in crossover.

Crucially, we will develop an explanation for crossover not in terms of *linear* order, but rather in terms of *evaluation* order. One reason linear order is not an adequate explanation is that there are systematic cases in which a quantifier can linearly follow a pronoun that it nevertheless can bind:

(6) a. Which of his$_i$ relatives does every man$_i$ love the most?
b. The relative of his$_i$ that every man$_i$ loves most is his mother.

We explain in some detail how our continuation-based approach accounts for these so-called reconstruction effects. In brief, the independently-motivated semantics of wh-fronting and relative clause formation *delay* the evaluation of the pronoun until after the evaluation of the quantifier. So on our continuation-based analysis, these exceptions follow automatically from the meaning of the expressions involved. The net prediction is that it is precisely in reconstruction cases that evaluation order comes apart from linear order.

In recognition of the importance of evaluation order to building a case that continuations are indispensable, Part I of the book is devoted to an in-depth case study of crossover and related phenomena, including in-situ wh and wh-fronting, donkey anaphora, coordination, and the order-sensitivity of negative polarity licensing. The analysis is expressed in a continuation-based grammar presented in what we call tower notation, as introduced in, e.g., Barker and Shan (2008). This formalism is expressly designed to be as easy to learn as possible, and in particular, easy to work with on paper and on a blackboard.

But a principled treatment of evaluation order is not the only distinctive advantage of taking a continuation-based view. Part II of the book investigates phenomena that do not depend on evaluation order. The first of the two main case studies involves scope-taking adjectives such as *same* and *different*.

(7) Ann and Bill read the same book.

The sentence-internal reading (the reading that does not depend on identifying some salient book from context) is notoriously difficult to treat compositionally (see, e.g., Carlson 1987, Keenan 1992). Following the parasitic-scope approach of Barker (2007), we show how a continuation-based analysis follows naturally from an analysis of nominal uses of *same*.

Because parasitic scope requires higher-order continuations (categories of the form $A\backslash\!\backslash(B\backslash\!\backslash C)$), which are beyond the expressive power of the grammar from Part I, we develop the analyses in Part II in a continuation-based type-logical grammar first introduced in Barker (2007).

The second main case study in Part II is sluicing.

(8) [John made someone happy], but I don't know who __.

The interpretation of the embedded interrogative *who* __ takes its meaning from the content of an antecedent clause, in this case, the bracketed initial clause. Furthermore, the wh-phrase *who* corresponds (in a sense made precise below) to the indefinite *someone*. In fact, the elided content is exactly the antecedent clause with the wh-correlate removed, namely, *John made* [ ] *happy*. But of course, this is exactly a continuation, namely, the continuation of *someone* relative to the bracketed antecedent clause. Following Barker (2013), we suggest that sluicing is anaphora to a continuation. If so, then we need a grammar that explicitly recognizes continuations, so that it can make them available to serve as potential antecedents.

The answer to the question "What makes continuations essential?", then, is that continuations enable new and potentially insightful analyses of crossover, reconstruction, NPI licensing, and other order-sensitive phenomena in terms of evaluation order; and that continuations provide robust, potentially insightful analyses of parasitic scope and sluicing.

A continuation is the rest of an expression, e.g., a scope remnant. In some sense, then, continuations are anti-constituents: the complement of an expression relative to some enclosing expression. Analyzing natural language without explicitly using continuations is like performing arithmetic without ever using negative numbers: many useful tasks can still be accomplished, but a full understanding requires taking a broader view.

Why are continuations so hard to understand?

Continuations have been studied in computer science, in logic, and in natural language semantics. No matter which discipline, they have a reputation for being hard to understand. Apparently, continuations are intrinsically hard to understand, at least at first.

We will spend considerable effort explaining what continuations are, concentrating especially on showing in detail how analyses that make use of continuations work on a practical level.

Furthermore, since no single presentation works best for everyone, we will come at continuations from several different directions. The majority of the book will be concerned with a grammar that we develop incrementally in Part I, with step-by-step explanations and many derivations and diagrams. Then we'll develop a different continuation-based grammar in Part II. The hope is that comparing these two different formalisms will clarify the essence of (one kind of) continuation.

In our experience, grasping continuations requires working with them. Therefore, we have provided a number of exercises throughout Part I, with complete solutions at the end of the book. Offering exercises is unusual for a research monograph in linguistics. We have included them with the encouragement of our editors, as well as of many of our students and colleagues. We hope that they're useful to some of our readers. If you aren't interested in doing exercises, you can think of them as a kind of footnote.

We can't promise that we will make continuations easy to understand. We can, however, promise three things. First, we promise we'll do our best to make understanding them as easy as possible. Second, we promise that, given a modest amount of effort on the part of the reader, it will be possible to understand in detail how a continuation-based analysis works. Finally, we promise that we'll do our best to make understanding continuations worth the effort, in terms of new insights into natural language semantics.

Audience

The level of the discussion in the first half of the book is aimed at readers who understand the standard tools and techniques of natural language semantics at the

"01-Barker-Prelims-drv" — 2014/9/8 — 13:42 — page xviii — #18

level of a first-year graduate student or of an advanced undergraduate. As usual, this amounts to some familiarity with phrase structure grammars, the first-order Predicate Calculus, the (simply-typed) lambda calculus, and, ideally, the basic ideas of generalized quantifiers in the spirit of Montague, as in, for example, Heim and Kratzer (1998).

Familiarity with techniques from the theory of programming languages will help, but is not necessary. van Eijck and Unger (2010) is a particularly congenial presentation of computational techniques in the service of semantic analysis.

Once the core of the approach has been established in the first several chapters, the theoretical and empirical investigations will begin to go deeper, and the level of the discussion will approach the usual level of a research monograph in terms of speed of presentation and assumptions about familiarity with standard literature.

Like any non-trivial formal technique, a complete understanding of continuations requires working through problems. We strongly urge the reader to complete a few key exercises on paper. The tower system has been designed specifically to make this both practical and rewarding.

We'll also do our best to help readers go beyond this operational-level understanding to a deeper appreciation, with comparisons with a number of other approaches, and pointers into the literature.

Ways to read this book

The heart of the book is an attempt to use continuations to gain a deeper understanding of natural language. The core ideas are presented in this introductory chapter, and in the two following chapters, chapters 1 and 2. These chapters develop continuations and the tower system up to a simple account of scope, binding, and crossover. Chapter 3 fits these ideas into a larger theoretical landscape, showing how taking a continuations-based perspective unifies Montague's generalized quantifier theory of DP meaning with the dynamic semantics perspective on sentence meaning, by treating them as two special cases of a more general interpretive strategy. If your goal is to merely get an impression of what continuations are and how they can be used to model natural language, these first four chapters will provide that much.

The remainder of Part I extends and deepens the core ideas and techniques both empirically and theoretically. The empirical extensions include reconstruction, order effects in negative polarity licensing, and donkey anaphora. The theoretical discussions include Partee and Rooth's (1983) theory of Generalized Coordination (section 7.1), Hendriks' (1993) Flexible Montague Grammar approach to scope-taking (section 7.2), Jacobson's (1999) Variable Free Semantics (sections 11.1 and 11.2), Steedman's (2012) theory of scope as surface constituency (section 11.3), and Plotkin's (1975) Continuation Passing-Style approach to evaluation order in the lambda calculus (section 12.1).

Part II analyzes sentence-internal uses of *same*, verb phrase ellipsis, and sluicing. Part II does not depend on Part I, and should be understandable even if it is read independently of Part I. Theoretical comparisons include Morrill's et al. (2011) Discontinuous Lambek Grammar (section 15.2), de Groote's (2001) application of the $\lambda\mu$-calculus to scope (section 18.1), and Bernardi and Moortgat's (2010) Lambek-Grishin calculus (section 18.2).

Historical notes

The research reported in this book began roughly at the turn of the millennium. In the case of Barker, this included a manuscript first circulated in 2000. An early version was published as Barker (2001), and a revised version was published as Barker (2002). In the case of Shan, many of the ideas developed in this book first appeared in Shan (2001a) and Shan (2001b), some of which are developed further in Shan (2005).

Continuations have been rediscovered many times in computer science, as related by Reynolds (1993), and the same is true in the study of natural language: we are by no means the first semanticists to make use of continuations. In fact, it is the burden of chapter 3 to argue that Montague's conception of DP meanings as generalized quantifiers is a form of continuation passing. Likewise, as that chapter also argues, we consider the dynamic conception of meaning (on which a sentence is a context update function, as in, e.g., Groenendijk and Stokhof 1990) as a different version of the same core idea of continuization. And yet again, we will suggest that Partee's (1987) treatment of generalized coordination (see section 7.1), as well as Hendriks' (1993) treatment of scope-taking (chapter 7.2) also make implicit use of continuations.

Nor are we the only researchers to study natural language semantics with explicit consideration of continuations. Around the same time we began our work, de Groote (2001) applied the $\lambda\mu$-calculus to natural language scope (section 18.1). He presented this work at the 2001 Amsterdam Colloquium, the same conference where Shan presented Shan (2001b). A few years later, de Groote (2006) gave a different continuation-based treatment of donkey anaphora (discussed in section 18.3). Among the growing list of continuation-based analyses of natural language, another one in particular that we will discuss below is Bernardi and Moortgat's (2010) Lambek-Grishin calculus (see section 18.2).

In other words, this book is neither the first word nor the last word on continuations in natural language. Our goal here is to explain what continuations are, how they work, and why they are potentially interesting to someone who studies the structure of language and meaning.

Part I

Towers: Scope and Evaluation Order

This first of the two main parts of the book presents a particular continuation-based grammar. The presentation and the grammar itself are designed to be as easy to learn and use as possible. The system is a combinatory categorial grammar with a small number of type-shifters, all of which apply freely and without constraint. Categories and semantic values are presented in a grid format we call "tower notation". Using this system, we incrementally develop a fragment that addresses quantifier scope, quantificational binding, dynamic anaphora, wh-fronting, relative clauses, and more.

The main explanatory goal is to provide a reconception of scope-taking. The main advantage for adopting a continuation perspective is that continuations allow fine-grained control over the order in which expressions are evaluated. If we assume that the order of evaluation defaults to left-to-right, we have an explanation for linear scope bias, as well as for crossover. At the same time, we show that various systematic exceptions to a simple left-to-right constraint on quantificational binding, in particular, certain reconstruction effects, fall out from independently-motivated assumptions about the meaning of the expressions involved.

Because the elements of the formal grammar are presented one by one, throughout the text, here is a complete list of where to find the introduction of each piece: the combination schema, (16); the four type-shifters, namely, LIFT (18), LOWER (21), BIND (29), and FRONT (62); the principle of applying type-shifters to subparts of categories, section 4.1; and the lexical schema for syntactic gaps, $A/\!\!/A$, section 5.1.

There is a compact description of a slightly refactored but equivalent grammar in section 12.2. (The modifications are motivated to make the grammar more computationally tractable.)

The first publication of an essentially equivalent formal grammar is Shan and Barker (2006); the first publication of the tower presentation is Barker and Shan (2008).

1

Scope and towers

This chapter gives a continuation-based grammar in the tower presentation, with just enough machinery to handle scope-taking. This will require defining a set of syntactic categories, and providing a way to combine expressions into complex expressions. In addition, there will be two complementary type-shifters, LIFT and LOWER. The next chapter adds binding, and the rest of Part I will build on the basic fragment, extending it when necessary, to handle a wide range of additional sentence types, concentrating on crossover and reconstruction.

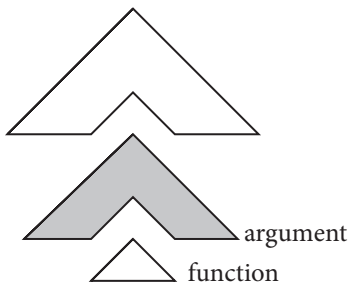
The fragment is a combinatory categorial grammar similar in nature to the systems in Jacobson (1999) or Steedman (2012), in which a small number of type-shifters ("combinators") apply freely and without constraint. It is faithful in spirit and in many details to the Shan and Barker (2006) analysis, though it uses the 'tower' notation introduced in Barker and Shan (2008).

1.1 Scope

Scope-taking is one of the most fundamental, one of the most characteristic, and one of the most dramatic features of the syntax and semantics of natural languages.

A phrase **takes scope** over a larger expression that contains it when the larger expression serves as the smaller phrase's semantic argument:

(9)



(10) John said [**Mary called** [everyone] **yesterday**] with relief.

In this schematic picture, the context *John said* [ ] *with relief* corresponds to the upper unshaded notched triangle, the embedded context *Mary called* [ ] *yesterday* corresponds to the middle gray notched triangle, and the scope-taker *everyone* corresponds to the lower unshaded triangle.

In (10), *everyone* takes scope over the rest of the embedded clause that surrounds it, namely, *Mary called* [ ] *yesterday*. Semantically, *everyone* denotes a function that takes as its argument the property $\lambda x.\textbf{yesterday}(\textbf{called}\ x)\ \ \textbf{m}$. We will call the expression over which the scope-taker takes scope (the gray region in the diagram) its **nuclear scope**.

The challenge for a theory of scope is to explain how it is possible for a scope-taker to reverse the direction of function/argument composition, that is, how it is possible to provide the scope-taking element with access to material that properly surrounds it.

The diagram of scope-taking is essentially identical to the diagram in (4) of the introductory chapter, the diagram that explained what a continuation is. This similarity is not accidental: the material over which a scope-taker takes scope is exactly what we are calling a continuation. This is what makes scope a particularly natural and compelling application for continuations in natural language.

Crucially, although all nuclear scopes can be viewed as continuations, the reverse is not true: the full range of continuations discussed in this book go beyond any standard analysis of scope-taking.

1.2 Syntactic categories: adjacency vs. containment

We'll need a set of syntactic category labels. This section develops a notation based on the categorial grammar tradition that we will use throughout the rest of the book.

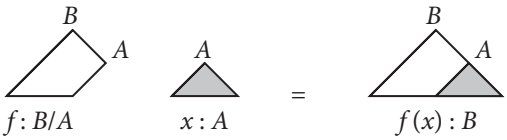
Normally, functors combine with arguments that are syntactically adjacent to them, either on the left or on the right. In the notation of categorial grammar (e.g., Lambek 1958), a functor in category $A\backslash B$ combines with an argument of category A to its left to form a B. So if *John* has category DP, and *left* has category DP\S, *John left* has category S.

(11)
$$\begin{pmatrix} \text{DP} & \text{DP}\backslash\text{S} \\ \textit{John} & \textit{left} \\ \textbf{j} & \textbf{left} \end{pmatrix} = \begin{matrix} \text{S} \\ \textit{John left} \\ \textbf{left}(\textbf{j}) \end{matrix}$$

Syntactically, we will call the operation that combines a functor with its argument 'merge'. Semantically, merge corresponds to function application, as usual. This means that the semantic type of an expression in category $A\backslash B$ will be $\alpha \rightarrow \beta$, where α is the type of expressions in category A, and β is the result type of the merged expression.

Schematically, for ordinary function application we have:

(12)

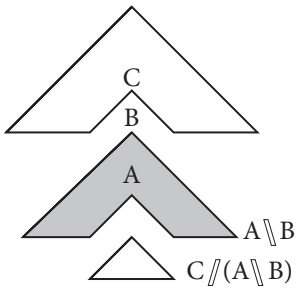


The functor category (here, B/A) is represented as a clear triangle with one corner missing: it would be a complete expression of category B if only we supply the missing corner of category A. The category of the argument (here, A) must match the category needed by the functor category, the category underneath the slash. This picture shows combination with a right-leaning slash, i.e., B/A instead of $A\backslash B$, as we had above in (11). The result after combination is a complete expression of category B.

For scope-taking, linear adjacency is not sufficient. After all, a scope-taker is not adjacent to its argument, it is contained within its argument. In (10), for instance, the scope-taker *everyone* is neither to the left or to the right of its nuclear scope—in fact, it's right in the middle. What we need, then, is a syntactic notion of 'surrounding' and 'being surrounded by'. Therefore we will augment the set of category labels with a second kind of slash: $A\backslash\!\backslash B$ and $B/\!/A$. On paper or on a blackboard, we write these symbols as double slash marks. Syntactically, we will see that these hollow slashes correspond to in-situ scope-taking.

Pursuing this idea, we will build up to a suitable category for a scope-taker such as *everyone* in two steps. First, consider again the schematic picture of scope-taking given above in (9), with some category labels added:

(13)



The category of the gray notched triangle in the middle—the nuclear scope—will be $A\backslash\!\backslash B$: something that would be a complete expression of category B, except that it is missing an expression of category A somewhere (specific) inside of it. Just like $A\backslash B$, $A\backslash\!\backslash B$ will have semantic type $\alpha \rightarrow \beta$: a function from objects of type α to objects of type β, assuming that α and β are the semantic types of expressions in categories A and B.

As may already be clear by now, an expression in a category of the form $A\backslash\!\backslash B$ will play the role of a continuation.

The second step in arriving at a category for a scope-taker is to consider the scope-taker itself, the small lower triangle in the diagram. It takes the continuation above it as its semantic argument. But once again, it is not adjacent to its argument. Rather, it is *surrounded* by its argument. Just as we needed a notion of 'missing something somewhere inside of it', we now need a notion of 'missing something surrounding it'. If $A \backslash\!\backslash B$ means 'something that would be a B if we could add an A somewhere (specific) inside of it', then we'll use $C /\!\!/ D$ to mean 'would be a C if we could add a D surrounding it'. Of course these two notions complement each other; and in fact, a little thought will reveal that the surrounding D will most naturally be a continuation, since continuations are the kind of expression that require something to be inserted inside of them.

The general form of a scope-taker, then, will be $C /\!\!/ (A \backslash\!\backslash B)$, as indicated in the diagram: something that combines with a continuation of category $A \backslash\!\backslash B$ surrounding it to form a complex expression in category C.

For example, consider the sentence *John called everyone yesterday*. The nuclear scope is the sentence missing the scope-taker: *John called* [ ] *yesterday*. This is an expression that would be an S except that it is missing a DP somewhere specific inside of it. So this continuation has category DP$\backslash\!\backslash$S. When the quantifier *everyone* combines with this continuation, it will form a complete sentence of category S. Therefore the syntactic category of the quantifier will be S$/\!\!/$(DP$\backslash\!\backslash$S): the kind of expression that needs a continuation of category DP$\backslash\!\backslash$S surrounding it in order to form a complete S.

1.3 Tower notation

A simple example of scope-taking will serve to introduce the tower notation:

(14)
$$\left(\begin{array}{cc} \frac{\mathrm{S}\,|\,\mathrm{S}}{\mathrm{DP}} & \frac{\mathrm{S}\,|\,\mathrm{S}}{\mathrm{DP}\backslash\mathrm{S}} \\ \textit{everyone} & \textit{left} \\ \frac{\forall y.\,[\,]}{y} & \frac{[\,]}{\textbf{left}} \end{array}\right) = \begin{array}{c} \frac{\mathrm{S}\,|\,\mathrm{S}}{\mathrm{S}} \\ \textit{everyone left} \\ \frac{\forall y.\,[\,]}{\textbf{left}\ y} \end{array}$$

There are several elements in this derivation that will be carefully explained in the course of the next few sections.

First, purely as a matter of notation, syntactic categories of the form $C /\!\!/ (A \backslash\!\backslash B)$ can optionally be written as $\frac{C\,|\,B}{A}$. This is what we call the 'tower' convention. The categories in this chapter have only two levels, and so don't really deserve to be called towers; however, in later chapters, categories will grow to include three or more layers.

Towers can be read counterclockwise, starting at the bottom: expressions in a category $\frac{C\,|\,B}{A}$ function locally (i.e., with respect to function/argument combination)

as an A, take scope over an expression of category B, and return a new expression of category C. So, as explained above, the syntactic category given here for *everyone* will be $S/\!\!/(DP\backslash\!\!\backslash S) \equiv \frac{S \mid S}{DP}$: something that functions locally as a DP, takes scope over an S, and returns as a result a new expression of category S.

Semantically, *everyone* will denote the usual generalized quantifier, namely, $\lambda\kappa.\forall y.\kappa y$, where κ is a variable of type $\mathtt{e} \rightarrow \mathtt{t}$. But, as illustrated above in (14), we can write semantic values in tower notation too:

(15)
$$\begin{array}{ccc} S/\!\!/(DP\backslash\!\!\backslash S) & & \frac{S \mid S}{DP} \\ \textit{everyone} & \equiv & \textit{everyone} \\ \lambda\kappa.\forall y.\kappa y & & \frac{\forall y.\,[\,]}{y} \end{array}$$

In general, a function of the form $\lambda\kappa.g[\kappa f]$ can optionally be written as $\frac{g[\,]}{f}$.

Here, $\forall y.[\,]$ and $g[\,]$ are *context*s, i.e., logical expressions containing a single hole. This notation is often seen in theoretical discussions of formal languages (for instance, Barendregt 1981: 29), including programming languages (e.g., Felleisen 1987), as well as in discussions of logical languages (for instance, in discussions of the cut rule in substructural logics, e.g., (Moortgat, 1997: 106) or (Restall, 2000: 112); see also chapter 13.

Contexts are not so familiar in linguistic discussions. However, because syntactic and semantic towers are completely equivalent with their flat (i.e., non-tower) counterparts, it is always possible to give a full analysis that does not rely on contexts that contain holes, if desired.

EXERCISE 1

What are the flat notational counterparts of the syntactic and semantic towers $\frac{S \mid S}{DP\backslash S}$ and $\frac{\forall x.[\,]}{\textbf{left}\ x}$?

1.4 The combination schema

Continuing the explanation of the derivation in (14), the combination of two multi-level towers cannot be simple functional application. Rather, it is described by the following schema:

(16) **The combination schema** ('/' variant):

$$\left(\begin{array}{cc} \dfrac{C \mid D}{B/A} & \dfrac{D \mid E}{A} \\ \text{left.exp} & \text{right.exp} \\ \dfrac{g[\,]}{f} & \dfrac{h[\,]}{x} \end{array}\right) = \begin{array}{c} \dfrac{C \mid E}{B} \\ \text{left.exp right.exp} \\ \dfrac{g[h[\,]]}{f(x)} \end{array}$$

On the syntactic tier, the horizontal line divides two different combination regimes. Beneath the horizontal line, B/A and A combine as usual to form a B; above the line, $C|D$ and $D|E$ combine to form $C|E$.

In parallel, on the semantic tier, below the horizontal line, combination is function application. Above the horizontal line, $g[\,]$ and $h[\,]$ combine to form $g[h[\,]]$. For instance, if $g[\,] = \forall x.[\,]$, and $h[\,] = \exists y.[\,]$, then $g[h[\,]] = \forall x[\exists y.[\,]]$.

EXERCISE 2

If $g[\,] = \lambda x.(x\,[\,])y$ and $g[h[\textbf{saw m}]] = \lambda x.(x(\textbf{thinks}(\textbf{saw m})))y$, what must $h[\,]$ be?

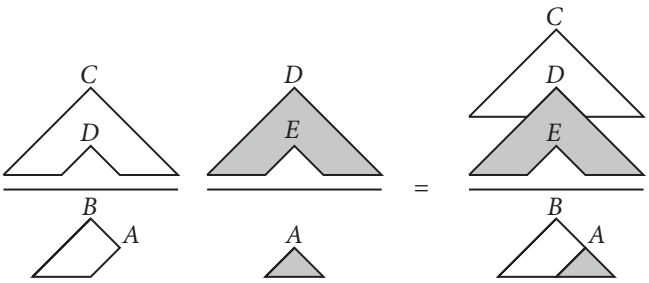
It is important to distinguish plugging a hole in a context from beta reduction in the lambda calculus. Unlike beta reduction, plugging a hole can result in variable capture. If $g[\,] = \lambda x.(x\,[\,])y$, then $g[x] = \lambda x.(xx)y$: plugging the hole in $g[\,]$ with x results in an expression in which the plug x is bound by the lambda.

Because the tower notation and flat notation are completely equivalent, if there is any uncertainty over the interaction of context holes with variable capture, in any concrete example it is always possible to translate the tower in question back into flat notation, and then make use of the usual rules of beta reduction.

The combination schema in (16) will be the general mode of composition that will be used throughout Part I. As we will explain in detail below, it embodies the general principle that expressions, by default, must be evaluated from left-to-right, so it is at the heart of our explanation of crossover, reconstruction, and other evaluation-order effects.

A tangram diagram will help unpack what is going on here conceptually.

(17)



Focusing first on the layer below the horizontal lines, the unshaded trapezoidal functor element has category B/A, and the small gray triangle has category A. They combine according to the normal function/argument pattern: the A categories match and cancel, forming an expression of category B. This is just diagram (12) repeated.

In the layer above the horizontal line, the syntactic categories D match (note that the labels in the diagram match the categories in the schema given above in (16) exactly), meaning that it is coherent to use the result of the notched gray continuation as the input to the unshaded continuation. This composition is indicated graphically on the right by inserting the gray subcomputation (E to D) into the notch of the unshaded computation (D to C). The result after combination is an expression that is waiting for a continuation that would fit in the space occupied by the horizontal line, namely, a computation that can turn a B into an E (i.e., a continuation of category $B\backslash\!\backslash E$). If such a continuation were supplied, the net result would be a complete expression of category C.

EXERCISE 3

The combination schema in (16) has a right-leaning slash in the syntactic category of its leftmost expression. What should the combination schema be when the two elements have categories $\frac{C \mid D}{A}$ and $\frac{D \mid E}{A \backslash B}$? It may help to draw the tangram diagram.

1.5 Types and continuations

The semantic types of the expressions in this grammar are straightforward. Expressions in category DP have semantic type `e`, the type of individuals, and expressions in category S have semantic type `t`, the type of truth values. As mentioned, expressions in category $A\backslash B$ have semantic type $\alpha \to \beta$, where α is the type of A and β is the type of B. Likewise, expressions in category $A\backslash\!\backslash B$ also have semantic type $\alpha \to \beta$, as do expressions in categories B/A and $B/\!\!/A$.

Thus since the syntactic category of *everyone* is $\frac{\mathrm{S} \mid \mathrm{S}}{\mathrm{DP}} \equiv \mathrm{S}/\!\!/(\mathrm{DP}\backslash\!\backslash\mathrm{S})$, the semantic type of *everyone* is $(\mathtt{e} \to \mathtt{t}) \to \mathtt{t}$. This is exactly what we expect for an extensional generalized quantifier.

And since the semantic value of *everyone* as given above in (14) is $\frac{\forall y.[\,]}{y} \equiv \lambda\kappa.\forall y.\kappa y$, the typing correspondence forces x to be a variable of type `e`, and κ to be a variable of type $\mathtt{e} \to \mathtt{t}$. In other words, as discussed in more depth in chapter (3), the continuation-based approach has led us more or less directly to the standard treatment of generalized quantifiers of Montague (1974) and Barwise and Cooper (1981).

Where in the semantics are the continuations? As suggested by the choice of variable symbol, κ stands for 'continuation'. For instance, in the case of *everyone*, the key fact is that the continuation of a DP relative to the clause it takes scope over will be a function of type $\mathtt{e} \rightarrow \mathtt{t}$, exactly the type we just assigned to κ.

1.6 The LIFT type-shifter

Another element that needs comment in the derivation in (14) above is that the syntax and semantics for *left* does not match the treatment it received in (11). The reason is that non-scope-taking elements such as *left* must be adjusted in order to combine with scope-takers. Just as Montague recognized that the denotations of proper names (which according to Partee (1987) are fundamentally of type e) must be adjusted in order to coordinate with properly quantificational DPs (type $(\mathtt{e} \rightarrow \mathtt{t}) \rightarrow \mathtt{t}$), so too must *left* be adjusted here. In both cases, the adjustment mechanism is the same: a generalization of Partee's (1987) LIFT type-shifter. In general, for all categories A and B, and for all semantic values x:

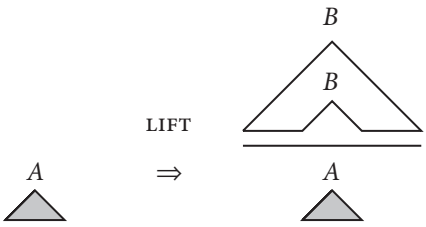
(18)

		$\frac{B \mid B}{}$
A	LIFT	A
phrase	$\Rightarrow$	*phrase*
x		$\frac{[\,]}{x}$

If x is the semantic value of an expression in category A, then the semantic value of the LIFTed A is $\frac{[\,]}{x} \equiv \lambda\kappa.\kappa x$.

The tangram version shows how LIFT turns a value into an expression that is expecting a continuation.

(19)



Looking at the result, the horizontal line represents the place where the continuation will fit. This expression expects a continuation that surrounds an expression of

category A to build an expression of category B. The idea of the LIFT operation is that it is easy for an expression of category A to use a continuation of category $A\backslash\!\backslash B$ to produce a result of category B: simply apply the continuation to the original expression of category A. In other words, the semantic content of the unshaded notched triangle above the horizontal line on the right is simply an identity function.

Two examples will serve to illustrate:

(20)

			$\frac{S \mid S}{DP}$					$\frac{S \mid S}{DP\backslash S}$
	DP	LIFT	DP			DP\S	LIFT	DP\S
(a)	*John*	⇒	*John*		(b)	*left*	⇒	*left*
	j		$\frac{[\,]}{\mathbf{j}}$			**left**		$\frac{[\,]}{\mathbf{left}}$

For instance, LIFTing the proper name *John* yields the usual generalized quantifier syntax and semantics, since $\frac{[\,]}{\mathbf{j}} \equiv \lambda\kappa.\kappa(\mathbf{j})$. Likewise, when the simple version of *left* given in (11) undergoes the LIFT typeshifter, the result is the verb phrase that appears above in the derivation of *everyone left* in (14).

Semantically, just as the generalized-quantifier version of *John* has no detectable scope-taking effect (it is 'scopeless'), so too the LIFTed *left* has no detectable scope-taking effect. This is evident from the fact that the semantics of the LIFT operator supplies just an empty context '[ ]' above the horizontal line (equivalent in flat notation to the identity function).

As chapter 3 will emphasize, continuizing throughout the grammar allows us to generalize Partee's LIFT type-shifter from something that originally only applied sensibly to expressions of type e to something that can apply to expressions in any category.

1.7 The LOWER type-shifter

The final element in the derivation of *everyone left* that requires explanation is the fact that the derivation as given above ends with a multi-level syntactic category. That is, the final syntactic category is $\frac{S \mid S}{S}$ instead of a plain S. A derivation like this would be appropriate if the clause were embedded in a larger expression over which the quantifier takes scope; but if we imagine that this is a complete utterance, we need a way to close off the scope domain of the quantifier. We accomplish this with the following type-shifter.

For all categories A, for all contexts $f[\ ]$, and for all semantic values x:

(21)

$\frac{A \mid \mathrm{S}}{\mathrm{S}}$	LOWER	A
phrase	$\Rightarrow$	*phrase*
$\frac{f[\ ]}{x}$		$f[x]$

If F is the semantic value of the original expression, then $F(\lambda\kappa.\kappa)$ is the value of the shifted expression. The idea is that the semantic tower is collapsed by plugging the hole in the context above the line with the material below the line:

(22)

This diagram makes it clear how it is possible to remove the horizontal line and collapse two levels into one, provided that the category of the plug (the lower gray element) matches the category expected by the context (the element above the line).

Crucially, the LOWER rule as given in (21) restricts the lowering operation to situations in which the category of the plug is S. Thus this lowering rule only applies to scope-taking elements that take scope over a clause. This limitation plays an important role in the explanation for crossover, as discussed in section 4.3.

Using the LOWER type-shifter, we can complete the derivation of *everyone left*:

(23)

$\frac{\mathrm{S} \mid \mathrm{S}}{\mathrm{S}}$	LOWER	S
everyone left	$\Rightarrow$	*everyone left*
$\frac{\forall y.\,[\ ]}{\textbf{left}\ y}$		$\forall y.\,\textbf{left}\ y$

The combinator lowers the category of the sentence back to a plain S.

Though lowering operations are not as common in the literature as lifting, the LOWER type-shifter plays a role here that is closely similar to Groenendijk and Stokhof's (1990) '↓' operator. We will comment further on this connection in section 3.3.

1.8 A linear scope bias

As we have mentioned, the explanation for crossover will depend heavily on imposing a left-to-right evaluation regime. This bias is already embodied in the combination schema given above in (16). As one symptom of the fact that the combination schema enforces left-to-right evaluation order, note that when a sentence contains two quantifiers, the quantifier on the left takes scope over the one on the right, at least as a default:

(24)
$$\begin{array}{c} \mathrm{S}\,|\,\mathrm{S} \\ \hline \mathrm{DP} \\ \textit{someone} \\ \exists x.\,[\,] \\ \hline x \end{array} \left(\begin{array}{c} \mathrm{S}\,|\,\mathrm{S} \\ \hline (\mathrm{DP}\backslash\mathrm{S})/\mathrm{DP} \\ \textit{loves} \\ [\,] \\ \hline \mathbf{loves} \end{array} \quad \begin{array}{c} \mathrm{S}\,|\,\mathrm{S} \\ \hline \mathrm{DP} \\ \textit{everyone} \\ \forall y.\,[\,] \\ \hline y \end{array} \right)$$

$$= \begin{array}{c} \mathrm{S}\,|\,\mathrm{S} \\ \hline \mathrm{S} \\ \textit{someone loves everyone} \\ \exists x.\,\forall y.\,[\,] \\ \hline \mathbf{loves}\; y\; x \end{array} \quad \overset{\textsc{lower}}{\Rightarrow} \quad \begin{array}{c} \mathrm{S} \\ \textit{someone loves everyone} \\ \exists x.\,\forall y.\,\mathbf{loves}\; y\; x \end{array}$$

This derivation involves two applications of the combination schema: once for the verb (the LIFTed *loves*) combining with its direct object (*everyone*), and once for the subject (*someone*) combining with the verb phrase.

> **EXERCISE 4**
>
> Give the intermediate result of combining *loves* with *everyone*.

As a result of the left-to-right bias built into the combination schema, this evaluates to $\exists x \forall y.\mathbf{loves}\, y\, x$, in which the existential quantifier takes linear (i.e., surface) scope over the universal.

Of course, we will also need a way to arrive at inverse scope; that will be postponed to chapter 4.

1.9 A scope ambiguity due to LOWER

In addition to its role as a derivation finisher, LOWER also serves as a way to delimit the region that a scope-taker takes scope over. Like all of the type-shifters in this book, LOWER is allowed to freely apply or not whenever its schema is matched. Whether

or not LOWER applies can determine which interpretation an ambiguous sentence receives.

(25) Mary wants everyone to leave.

For instance, if LOWER applies to the embedded clause *everyone to leave* before it combines with the matrix verb *wants*, then the scope of the quantifier will be limited to the embedded clause, and the interpretation of the sentence will be $\mathbf{wants}(\forall x.\mathbf{leave}\ x)\ \mathbf{m}$. In this interpretation, Mary has a single desire: that everyone leave.

But if LOWER does not apply until the end of the derivation, a different interpretation emerges:

(26)
$$\begin{array}{c} \mathrm{S}\,|\,\mathrm{S} \\ \hline \mathrm{DP} \\ \textit{Mary} \\ [\ ] \\ \hline \mathbf{m} \end{array} \left(\begin{array}{c} \mathrm{S}\,|\,\mathrm{S} \\ \hline (\mathrm{DP}\backslash\mathrm{S})/\mathrm{S} \\ \textit{wants} \\ [\ ] \\ \hline \mathbf{wants} \end{array} \left(\begin{array}{c} \mathrm{S}\,|\,\mathrm{S} \\ \hline \mathrm{DP} \\ \textit{everyone} \\ \forall x.[\ ] \\ \hline x \end{array} \begin{array}{c} \mathrm{S}\,|\,\mathrm{S} \\ \hline \mathrm{DP}\backslash\mathrm{S} \\ \textit{to leave} \\ [\ ] \\ \hline \mathbf{leave} \end{array} \right)\right)$$

$$= \begin{array}{c} \mathrm{S}\,|\,\mathrm{S} \\ \hline \mathrm{S} \\ \textit{M.w.e.t.l} \\ \forall x.[\ ] \\ \hline \mathbf{wants}(\mathbf{leave}\ x)\ \mathbf{m} \end{array} \overset{\text{LOWER}}{\Rightarrow} \begin{array}{c} \mathrm{S} \\ \textit{Mary wants everyone to leave} \\ \forall x.\mathbf{wants}(\mathbf{leave}\ x)\ \mathbf{m} \end{array}$$

In this interpretation, Mary has a multiplicity of desires: for each person x, Mary wants x to leave.

This ability of LOWER to encapsulate scope within a circumscribed domain is closely analogous to what Danvy and Filinski (1990) call a 'reset' in their theory of layered continuations. It is also similar to the mechanism for limiting scope in Barker (2002). See Charlow (2014) for discussion of reset as a mechanism for enforcing a general theory of scope islands in the context of a continuation-based grammar.

We now have a grammar that allows scope-taking elements to take scope over a portion of the material that surrounds them. At this point, we can handle sentences with multiple quantifiers, though we are for the moment limited to linear scope. We can also explain some scope ambiguities, if the scope-taking element interacts with some other element in the sentence (here, the verb *wants*).

2

Binding and crossover

When a quantifier follows a pronoun it is trying to bind, the usual result is a mild kind of ungrammaticality known as a crossover violation.

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

As discussed in the introduction, a quantifier can bind a pronoun that follows it, as in (27a), but usually not when the pronoun precedes it, as in (27b). In this chapter, we will extend the grammar to account for this basic contrast, and then build a progressively more refined account of the phenomenon in later chapters.

In order to discuss crossover, we will first need to provide a way for quantifiers to bind pronouns. This will involve adding a new connective to the syntactic categories: $A \triangleright B$ will be a syntactic category whenever A and B are syntactic categories. Its semantic type will be $\alpha \to \beta$, where α and β are the types of A and B. This will be the category of something that is basically an expression in category B, but that contains a pronoun of category A needing to be bound.

Just as in Jacobson (1999), the presence of an unbound pronoun will be recorded on the category of each larger expression that contains it. In particular, a clause containing an unbound pronoun will have category DP $\triangleright$ S rather than plain S. In order to accomplish this, the essential assumption here is that a pronoun must take scope, as advocated by, e.g., Dowty (2007): it will function locally as a DP, take scope over an S, and turn that S into an open proposition:

(28)
$$\left(\begin{array}{cc} \frac{\text{DP} \triangleright \text{S} \,|\, \text{S}}{\text{DP}} & \frac{\text{S} \,|\, \text{S}}{\text{DP}\backslash\text{S}} \\ \textit{he} & \textit{left} \\ \frac{\lambda y.\,[\,]}{y} & \frac{[\,]}{\textbf{left}} \end{array}\right) = \frac{\frac{\text{DP} \triangleright \text{S} \,|\, \text{S}}{\text{S}}}{\begin{array}{c}\textit{he left}\\ \frac{\lambda y.\,[\,]}{\textbf{left}\ y}\end{array}} \quad \textsc{lower} \quad \begin{array}{c}\text{DP} \triangleright \text{S}\\ \Rightarrow\\ \end{array} \quad \begin{array}{c}\textit{he left}\\ \lambda y.\,\textbf{left}\ y\end{array}$$

Note that the lexical denotation of the pronoun, $\frac{\lambda y.\,[\,]}{y} \equiv \lambda\kappa y.\kappa y$, is a (two-place) identity function.

On this view, sentences with free pronouns do not have the same category as sentences without pronouns (e.g., *John left*). This difference reflects the fact that a

sentence containing a pronoun means something different from an ordinary sentence: it does not express a complete thought until the value of the pronoun has been specified, whether by binding or by the pragmatic context. However, the two sentence types still share a common core; this commonality is captured here by the fact that in the tower notation, below the line, they are both S's, and it is only above the line that their differences are marked.

Once a pronominal dependency has been created, how will it be satisfied? Certainly, it must be possible for the value of the embedded pronoun to be supplied by the pragmatic context. But it must also be possible for some other element within the utterance to control (i.e., to bind) the value of the pronoun. We accomplish this by providing a type shifter called BIND, which enables an arbitrary DP to bind a downstream pronoun. For any categories A and B:

(29)

$\frac{A \mid B}{\text{DP}}$	BIND	$\frac{A \mid \text{DP} \rhd B}{\text{DP}}$
phrase	$\Rightarrow$	*phrase*
$\frac{f[\ ]}{x}$		$\frac{f[[\ ]\,x]}{x}$

Intuitively, this type-shifter feeds a copy of x to the function that will be used to plug the hole in $f[\ ]$. For example:

(30)

$\frac{\text{S} \mid \text{S}}{\text{DP}}$	BIND	$\frac{\text{S} \mid \text{DP} \rhd \text{S}}{\text{DP}}$
everyone	$\Rightarrow$	*everyone*
$\frac{\forall x.\,[\ ]}{x}$		$\frac{\forall x.[\ ]x}{x}$

The shifted expression has category $\text{S}/\!\!/(\text{DP}\backslash\!\!\backslash(\text{DP} \rhd \text{S}))$ and semantics $\lambda\kappa\forall x.\kappa\,x\,x$: something that knows how to turn a surrounding sentence containing a pronoun ($\text{DP} \rhd \text{S}$) into a plain clause (S). It accomplishes this semantically by quantifying over individuals ($\forall x$), and then applying the continuation κ to two copies of each individual x ($\kappa\,x\,x$), rather than to just one.

We immediately have an account of quantificational binding:

(31)

$$\frac{\text{S} \mid \text{DP} \rhd \text{S}}{\begin{array}{c}\text{DP}\\ \textit{everyone}\\ \forall x.[\ ]x\\ \hline x\end{array}} \left(\frac{\text{DP} \rhd \text{S} \mid \text{DP} \rhd \text{S}}{\begin{array}{c}(\text{DP}\backslash\text{S})/\text{DP}\\ \textit{loves}\\ [\ ]\\ \hline \textbf{loves}\end{array}} \left(\frac{\text{DP} \rhd \text{S} \mid \text{S}}{\begin{array}{c}\text{DP}\\ \textit{his}\\ \lambda y.\,[\ ]\\ \hline y\end{array}} \quad \frac{\text{S} \mid \text{S}}{\begin{array}{c}\text{DP}\backslash\text{DP}\\ \textit{mother}\\ [\ ]\\ \hline \textbf{mom}\end{array}} \right) \right)$$

$$\begin{array}{ccccc} & \dfrac{\mathrm{S}\,|\,\mathrm{S}}{\mathrm{S}} & & & \\ = & \textit{Everyone loves his mother} & \textsc{lower} & \textit{Everyone loves his mother} & \\ & \dfrac{\forall x.(\lambda y.\,[\,])x}{\textbf{loves}\ (\textbf{mom}\ y)\ x} & \Rightarrow & \forall x.(\lambda y.\textbf{loves}\ (\textbf{mom}\ y)\ x)x & \end{array}$$

Note that the syntactic tower for *loves* gets its upper layer by choosing $B = \mathrm{DP} \rhd \mathrm{S}$ when applying LIFT. After beta reduction (i.e., lambda conversion), the semantic value is $\forall x.\ \textbf{loves}\ (\textbf{mom}\ x)\ x$. This is an interpretation on which the quantifier binds the pronoun, as desired.

EXERCISE 5

Compute the functions denoted by *his mother* and by *loves his mother*.

2.1 The irrelevance of c-command

Since c-command has no special status in our theory of binding, it is perfectly possible to have binding without c-command.

(32)
$$\left(\begin{array}{cc} \dfrac{\mathrm{S}\,|\,\mathrm{DP} \rhd \mathrm{S}}{\mathrm{DP}} & \dfrac{\mathrm{DP} \rhd \mathrm{S}\,|\,\mathrm{DP} \rhd \mathrm{S}}{\mathrm{DP}\backslash\mathrm{DP}} \\ \text{everyone's} & \text{mother} \\ \dfrac{\forall x.[\,]x}{x} & \dfrac{[\,]}{\textbf{mother}} \end{array}\right)\left(\begin{array}{cc} \dfrac{\mathrm{DP} \rhd \mathrm{S}\,|\,\mathrm{DP} \rhd \mathrm{S}}{(\mathrm{DP}\backslash\mathrm{S})/\mathrm{DP}} & \dfrac{\mathrm{DP} \rhd \mathrm{S}\,|\,\mathrm{S}}{\mathrm{DP}} \\ \text{loves} & \text{him} \\ \dfrac{[\,]}{\textbf{loves}} & \dfrac{\lambda y.[\,]}{y} \end{array}\right)$$

The final interpretation is equivalent to $\forall y.\ \textbf{loves}\ y\ (\textbf{mom}\ y)$. The quantifier is able to bind the pronoun even though the quantifier is embedded in possessor position inside the subject, and so therefore does not c-command the pronoun.

One way to see what is going on here is to consider the upper level of the syntactic towers in (32): there is a chain of matching $\mathrm{DP} \rhd \mathrm{S}$'s connecting the possessor with the pronoun, and this chain is not disrupted by the major constituent boundary between the subject and the verb phrase. Put another way, it is often possible to ignore syntactic constituency when computing the upper levels of a tower.

Allowing a quantifier to bind a pronoun without c-commanding it is unorthodox. Since Reinhart (1983), it is almost universally assumed that quantificational binding requires c-command. Indeed, in textbooks such as Heim and Kratzer (1998:261) and Büring (2005:91), the very definition of binding requires the binder to c-command to bindee. Notable dissenters include Bresnan (1994, 1998), Safir (2004a,b), and Jäger (2005). Building on Shan and Barker (2006), Barker (2012) makes a case in some detail that there is abundant empirical motivation for rejecting this requirement.

Here is a representative sample of the data discussed in Barker (2012):

(33) a. [Everyone$_i$'s mother] thinks he$_i$'s a genius.
b. [Someone from every$_i$ city] hates it$_i$.
c. John gave [to each$_i$ participant] a framed picture of her$_i$ mother.
d. We [will sell no$_i$ wine] before it$_i$s time.
e. [After unthreading each$_i$ screw], but before removing it$_i$ . . .
f. The grade [that each$_i$ student receives] is recorded in his$_i$ file.

This data shows that quantifiers can bind pronouns even when the quantifier is embedded in a possessive DP, in a nominal complement, in a prepositional phrase, in a verb phrase, in a temporal adjunct, even when embedded inside of a relative clause. In each example, the quantifier does not c-command the pronoun. Various modifications and extensions of c-command have been proposed to handle some of the data, but Barker (2012) argues that none of these redefinitions covers all of the data.

Furthermore, as the derivation in (32) demonstrates, it is perfectly feasible to build a grammar in which a quantifier binds a pronoun without c-commanding it. Nothing special needs to be said; indeed, we would need to take special pains to impose a c-command requirement. In view of the data and the discussion in Barker (2012), then, we will assume that there is no c-command requirement on quantificational binding.

There is, of course, one important class of examples where a c-command restriction on quantificational binding makes good predictions, namely, crossover configurations. Any analysis that tries to do without a c-command restriction must supply an explanation for crossover that does not depend on c-command. This is exactly what we will do, in considerable detail, starting with the next sections, and continuing in later chapters.

2.2 The standard approach to crossover

The name "crossover" comes from Postal's (1971:62) proposal for a general constraint on movement, roughly: a DP may not move across a coindexed pronoun. This prohibition is usually implemented by taking advantage of multiple stages for the derivation of a sentence. Reinhart's (1983) is an influential example, and Büring (2001, 2004) provides a more recent analysis that uses the same basic strategy. The idea is to postulate two distinct levels of representation: first, a level of surface structure at which binding is established by some syntactic relationship based on c-command, then a level of Logical Form at which quantifiers take their semantic scope. This way, even though a quantifier may raise at LF to take scope over a pronoun, it can nevertheless only bind the pronoun if it c-commands the pronoun from its surface position.

Of course, this strategy is only available if binding depends on c-command. Because we reject c-command as a requirement for quantificational binding for the reasons given in the previous section, we cannot adopt the traditional strategy.

What, then, determines crossover? A simple leftness condition (linear precedence) would cover some of the data, but, as mentioned in the introduction, there are systematic counterexamples involving reconstruction, repeated here from (6):

(34) a. Which of his$_i$ relatives does every man$_i$ love the most?
b. The relative of his$_i$ that every man$_i$ loves most is his mother.

On the standard approach, reconstruction examples require syntactically moving some material, including the pronoun, back into the gap position before checking compliance with the syntactic c-command requirement, so that crossover violations can only be assessed at a strictly intermediate stage of the derivation.

The explanation developed here is that quantificational binding depends not on c-command or on simple leftness, but rather, on evaluation order.

2.3 A first crossover example

Because continuations are well-suited for reasoning about evaluation order (see the discussion in section 12.1), they enable us to explain crossover as an effect of the order in which the various elements of the sentence are processed (evaluated).

Here is what happens in a classic weak crossover configuration, i.e., when we try to allow a quantifier to bind a pronoun when the quantifier follows the pronoun.

(35)
$$\left(\begin{array}{cc} \dfrac{\mathrm{DP} \rhd \mathrm{S} \mid \mathrm{S}}{\mathrm{DP}} & \dfrac{\mathrm{S} \mid \mathrm{S}}{\mathrm{DP}\backslash\mathrm{DP}} \\ \text{his} & \text{mother} \end{array}\right)\left(\begin{array}{cc} \dfrac{\mathrm{S} \mid \mathrm{S}}{(\mathrm{DP}\backslash\mathrm{S})/\mathrm{DP}} & \dfrac{\mathrm{S} \mid \mathrm{DP} \rhd \mathrm{S}}{\mathrm{DP}} \\ \text{loves} & \text{everyone} \end{array}\right)$$
$$= \frac{\mathrm{DP} \rhd \mathrm{S} \mid \mathrm{DP} \rhd \mathrm{S}}{\mathrm{S}}$$
his mother loves everyone

Combination proceeds smoothly, and the complete string is recognized as a syntactic (and semantic) constituent. However, the result is not a complete derivation of a clause. In particular, it can't be lowered, since the category of the expression does not match the input to the LOWER type-shifter. The reason is that LOWER requires the subcategories in the upper right corner of the tower (here, DP $\rhd$ S) and beneath the horizontal line (S) to match. Even more restrictively, it requires these subcategories to be S. (See section 4.3 for a discussion of the explanatory status of these syntactic restrictions.) This means that in the derivation in (35), the pronoun continues to need a binder, and the quantifier continues to need something to bind.

EXERCISE 6

Compute the semantic value of the expression derived in (35).

Thus even though the quantifier takes scope over the entire clause, it is unable to bind a pronoun that precedes it.

We will discuss additional examples of crossover, as well as apparent exceptions to crossover, including reconstruction, in the chapters to follow.

2.4 Strong crossover

The derivation just shown concerns weak crossover, in which the pronoun to be bound does not c-command the quantifier in question. Situations in which the pronoun does c-command the quantifier are know as strong crossover. There is a qualitative difference between strong crossover and weak crossover.

(36) a. *He$_i$ loves everyone$_i$.
b. ?His$_i$ mother loves everyone$_i$.

The standard judgment is that weak crossover examples can be interpreted with some effort, but that strong crossover examples are irredeemable. (We will speculate about what might be going on when a comprehender makes the effort to interpret a weak crossover example in the next section.)

Although the account here rules out weak crossover and strong crossover alike, nothing in the formal system as presented distinguishes strong crossover from weak crossover. Presumably, strong crossover is due to some factor over and above whatever characterizes weak crossover, perhaps something along the lines of Safir's (2004b) (chapter 3) Independence Principle. The Independence Principle entails that a pronoun must not c-command anything that binds it.

2.5 Reversing the order of evaluation

In order to explore the role that order of evaluation plays in the crossover explanation, we can temporarily replace the combination schema given above in (16) with a variant that imposes a right-to-left processing bias.

(37)
$$\left(\begin{array}{cc} \frac{D \mid E}{A/B} & \frac{C \mid D}{B} \\ \textit{left} & \textit{right} \\ \frac{g[\,]}{f} & \frac{h[\,]}{x} \end{array}\right) = \begin{array}{c} \frac{C \mid E}{A} \\ \textit{left right} \\ \frac{h[g[\,]]}{f(x)} \end{array}$$

Here, it is the *outer* corner categories of the syntactic towers that must match (the *D*'s). On the semantic tier, now it is the rightmost context ($h[\,]$) that takes scope over the leftmost context ($g[\,]$).

EXERCISE 7

Draw the tangram diagram similar to (17) for the right-to-left variant of the combination schema as given in (37).

We can illustrate how this variant combination schema delivers reversed evaluation order bias by showing a derivation of *Someone loves everyone*:

(38)
$$\dfrac{\dfrac{\mathrm{S}\,|\,\mathrm{S}}{\mathrm{DP}}}{\begin{array}{c}\textit{someone}\\ \exists x.\,[\,]\\ \hline x\end{array}} \left(\dfrac{\dfrac{\mathrm{S}\;|\;\mathrm{S}}{(\mathrm{DP}\backslash\mathrm{S})/\mathrm{DP}}}{\begin{array}{c}\textit{loves}\\ [\,]\\ \hline \mathbf{loves}\end{array}} \quad \dfrac{\dfrac{\mathrm{S}\,|\,\mathrm{S}}{\mathrm{DP}}}{\begin{array}{c}\textit{everyone}\\ \forall y.\,[\,]\\ \hline y\end{array}} \right)$$

$$\dfrac{\dfrac{\mathrm{S}\,|\,\mathrm{S}}{\mathrm{S}}}{\begin{array}{c}\textit{Someone loves everyone}\\ \forall y\,\exists x.\,[\,]\\ \hline \mathbf{loves}\;y\;x\end{array}} \quad \textsc{lower} \quad \dfrac{\mathrm{S}}{\begin{array}{c}\textit{Someone loves everyone}\\ \forall y.\,\exists x.\,\mathbf{loves}\;y\;x\end{array}}$$

Using the variant combination schema in (37), the key thing to note is that the universal quantifier now takes scope over the existential, that is, the variant delivers inverse scope by default.

Crucially here, the right-to-left combination schema also enables a quantifier to bind a pronoun that precedes it:

(39)
$$\dfrac{\dfrac{\mathrm{DP}\,\triangleright\,\mathrm{S}\,|\,\mathrm{S}}{\mathrm{DP}}}{\begin{array}{c}\textit{he}\\ \lambda x.\,[\,]\\ \hline x\end{array}} \left(\dfrac{\dfrac{\mathrm{DP}\,\triangleright\,\mathrm{S}\,|\,\mathrm{DP}\,\triangleright\,\mathrm{S}}{(\mathrm{DP}\backslash\mathrm{S})/\mathrm{DP}}}{\begin{array}{c}\textit{loves}\\ [\,]\\ \hline \mathbf{loves}\end{array}} \quad \dfrac{\dfrac{\mathrm{S}\,|\,\mathrm{DP}\,\triangleright\,\mathrm{S}}{\mathrm{DP}}}{\begin{array}{c}\textit{everyone}\\ \forall y.\,([\,]\;y)\\ \hline y\end{array}} \right)$$

$$\dfrac{\dfrac{\mathrm{S}\,|\,\mathrm{S}}{\mathrm{S}}}{\begin{array}{c}\textit{He loves everyone}\\ \forall y\,(\lambda x.\,([\,]\;y))\\ \hline \mathbf{loves}\;y\;x\end{array}} \quad \textsc{lower} \quad \dfrac{\mathrm{S}}{\begin{array}{c}\textit{He loves everyone}\\ \forall y.\,(\lambda x.\,\mathbf{loves}\;y\;x)\;y\end{array}}$$

The way in which the alternative combination schema incorrectly allows the derivation of this ungrammatical crossover interpretation is by requiring the *outer* corners of the syntactic towers to match, not the inner corners, as we have been assuming so

far. Because of this, the BIND type-shifter and the LOWER type-shifter do not need to be adjusted in order for this crossover derivation to go through. After lambda reduction, the final result is $\forall y.$**loves** $y\ y$.

Incidentally, if there is an independent prohibition against a pronoun c-commanding a binder, this strong crossover example will be correctly ruled out, but the right-to-left schema would still incorrectly derive weak crossover examples such as ?*His$_i$ mother loves everyone$_i$*.

EXERCISE 8

Show that with only the right-to-left version of combination available, it is no longer possible to derive *Everyone$_i$ loves his$_i$ mother*.

2.6 Default evaluation order is left-to-right

Following Shan and Barker (2006), in view of the bias for linear (surface) quantifier scope, as well as the ungrammaticality of crossover interpretations, we will adopt the following hypothesis:

(40) By default, natural language expressions are processed from left-to-right.

Here, "left-to-right" means the temporal order in which expressions are produced and perceived. We will implement this hypothesis by assuming that the only combination schema available for normal processing is the left-to-right schema given above in (16).

Now, precisely because weak crossover is weak, it is possible to overcome the default bias and find an interpretation for a crossover example. Shan and Barker (2006) suggest that one possible explanation for this might be that if pressed by pragmatic context, a comprehender can exceptionally resort to a right-to-left evaluation schema. (Strong crossover would continue to be fully ungrammatical by virtue of violating some separate requirement such as Safir's Independence Principle.)

There are other possible explanations. For instance, we will consider a second possible explanation involving a variant of the LOWER type-shifter below in section 4.3.

In any case, we have seen that replacing the combination schema with a variant creates multiple effects related to a global reversal of the default order of evaluation, including default inverse scope and crossover amelioration. At the very least, this shows how continuations provide a principled way to reason about order.

We will continue to develop our account of crossover and its exceptions step by step throughout Part I, giving special attention to cases in which evaluation order is correctly predicted to diverge from linear order. For example, in reconstruction examples such as *the relative of his$_i$ that everyone$_i$ loves the most*, a pronoun linearly precedes the quantifier that binds it.

"03-Barker-c02-drv" — 2014/9/8 — 10:42 — page 23 — #9

But the key ingredients of our explanation are already in place: we have a continuation-based system that allows scope-taking expressions such as *everyone* to take scope over a portion of a larger expression, a mechanism for DPs to bind pronouns, and a default left-to-right bias in composition that, in simple examples, prevents a quantifier from binding a pronoun that precedes it.

3

From generalized quantifiers to dynamic meaning

The last two chapters presented a basic continuation-based grammar that handled a limited form of scope-taking and binding, including some simple examples of crossover. The empirical robustness of the approach will be addressed in later chapters, but in the meantime this chapter pauses to comment on what is going on conceptually.

More specifically, we'll show how taking a continuation-based perspective enables us to unify two fundamental breakthroughs in semantics that might otherwise seem independent: Montague's conception of DPs as generalized quantifiers on the one hand, and the central idea of dynamic semantics on the other hand, namely, the conception of sentences as update functions on their contexts. Here's how: we've already seen that generalized quantifiers are functions on DP continuations. We shall see that dynamic sentence meanings are functions on S continuations. Because a continuation-based grammar provides access to continuations systematically to every expression type, explicit use of continuations allows us to recognize these two major insights as special cases of a single more general strategy.

3.1 Capturing the duality of DP meaning

The puzzle is a familiar one. Proper names such as *John*, *Mary*, and *Bill* behave syntactically (almost) exactly like quantificational expressions such as *everyone*, *someone* and *no one*: they can all serve as subjects, direct objects, indirect objects, they can all passivize, participate in raising constructions, and so on. Yet their internal semantics is radically different: names (on some accounts) refer to entities, but the quantificational expressions do not refer at all; rather, they quantify over some class of entities. This contrast gives rise the following question:

(41) **Duality of DP meaning**: What (if anything) unifies the meanings of quantificational versus non-quantificational DPs?

The answer here will be that DPs uniformly have access to their continuations (just like any expression type). The difference between a name and a quantificational DP is that the quantificational DP makes non-trivial use of its continuation.

"04-Barker-c03-drv" — 2014/9/8 — 10:43 — page 25 — #2

Once we have an answer to the duality question, we can go on to ask the following questions:

(42) **Scope displacement**: Why does the semantic scope of a quantificational DP sometimes differ from its syntactic scope?

The answer that we will give here depends on the fact that continuations deliver surrounding semantic context up to some larger enclosing constituent (typically, a clause). As a result, supposing that quantifiers denote functions on their own continuation entails that they can take scope over a larger expression.

What do other theories have to say about these two questions?

We take Quantifier Raising (QR) at a level of Logical Form (LF) to be the dominant view of natural language quantification among linguists and perhaps among philosophers, or at the very least, the most universally familiar one. The QR story is enormously persuasive and robust, both from a descriptive and from an explanatory point of view. For the sake of concreteness, we will use Heim and Kratzer (1998) (see especially their chapters 6 and 7) as our reference version for the standard QR view.

On Heim and Kratzer's standard version of the story, non-quantificational DPs denote entities (type $\mathtt{e}$). Quantificational DPs (henceforth, 'QPs') denote generalized quantifiers (type $(\mathtt{e} \rightarrow \mathtt{t}) \rightarrow \mathtt{t}$), and typical transitive verbs denote relations over entities (type $\mathtt{e} \rightarrow \mathtt{e} \rightarrow \mathtt{t}$). Heim and Kratzer assume that composition is driven by types: the direction of function/argument combination is determined by whichever of the constituents has a type that is a function on objects corresponding to the type of the other constituent. Thus when a QP occurs in subject position, type-driven composition allows (indeed, requires) it to take the verb phrase (type $\mathtt{e} \rightarrow \mathtt{t}$) as an argument. However, when a QP occurs in a non-subject position, including direct object position, a type mismatch occurs: neither the verb nor the quantificational object denotes a function of the right type to take the other as an argument. Since interpretation would otherwise be impossible, QR moves the offending QP to adjoin higher in the tree, leaving behind a bound trace of type $\mathtt{e}$ in the original DP position, and triggering a special interpretation rule called Predicate Abstraction. These adjustments repair the type mismatch, and simultaneously explain scope displacement.

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

"04-Barker-c03-drv" — 2014/9/8 — 10:43 — page 26 — #3

Choosing the other horn of the dilemma, Montague (1974) gives a compelling answer to the duality question: non-quantificational DPs and QPs all denote generalized quantifiers. That is why they have closely similar syntactic distribution. Indeed, in the PTQ fragment, predicates accept generalized quantifiers in any NP position without any type mismatch.

But nothing in the type system forces QPs to take wide scope. As a result, Montague needs to stipulate a separate operation of Quantifying In to account for scope displacement. Once again we fail to provide a unified answer to both questions.

Therefore consider the following proposal:

(43) **The continuation hypothesis** (repeated from (1)): some natural language expressions denote functions on continuations, i.e., on their own semantic context.

In particular, assume that QPs denote functions on their continuations.

With respect to DP duality, QPs are just the continuation-aware version of non-quantificational DPs. Once the entire grammar is continuized, there is no type clash when they occur in object position, or in any other DP argument positions, since we can freely LIFT non-quantificational DPs so that they function as generalized quantifiers. To the extent that the availability of LIFT is stipulated, duality may be less than inevitable here; though see the type-logical treatment in Part II, on which LIFT is a theorem, and does not need to be stipulated.

As for scope displacement, because of the nature of continuations, merely stating the truth conditions of a QP in terms of continuations automatically guarantees that it will have semantic scope over an entire clause—in other words, scope displacement follows directly from the semantic nature of quantification. In sum, both the duality of DP meaning and scope displacement follow from the single assumption that determiner phrase meanings have access to their continuations.

3.2 Deriving context update functions

If we continuize uniformly throughout the grammar, then we will continuize both the category DP and the category S in one move. But saying that a clause denotes a function on its continuation amounts to deducing one of the core ideas of dynamic semantics.

We can show what we have in mind by giving an analysis of a simple discourse. If we make the usual assumption that a sequence of declarative sentences can be interpreted via (ordinary, non-dynamic) conjunction (category (S\S)/S), we immediately have an analysis of the mini-discourse *Someone$_i$ entered. He$_i$ left*:

(44)
$$\left(\begin{array}{cc} \dfrac{\mathrm{S}\,|\,\mathrm{DP}\rhd\mathrm{S}}{\begin{array}{c}\mathrm{DP}\\ \textit{someone}\\ \exists y.\,([\,]\,y)\end{array}} & \dfrac{\mathrm{DP}\rhd\mathrm{S}\,|\,\mathrm{DP}\rhd\mathrm{S}}{\begin{array}{c}\mathrm{DP}\backslash\mathrm{S}\\ \textit{entered}\\ [\,]\end{array}} \end{array}\right)\left(\begin{array}{c} \dfrac{\mathrm{DP}\rhd\mathrm{S}\,|\,\mathrm{DP}\rhd\mathrm{S}}{\begin{array}{c}(\mathrm{S}\backslash\mathrm{S})/\mathrm{S}\\ \textit{[period]}\\ [\,]\end{array}} \end{array}\right)\left(\begin{array}{cc} \dfrac{\mathrm{DP}\rhd\mathrm{S}\,|\,\mathrm{S}}{\begin{array}{c}\mathrm{DP}\\ \textit{he}\\ \lambda x.[\,]\end{array}} & \dfrac{\mathrm{S}\,|\,\mathrm{S}}{\begin{array}{c}\mathrm{DP}\backslash\mathrm{S}\\ \textit{left}\\ [\,]\end{array}} \end{array}\right)$$

$$\dfrac{\dfrac{\dfrac{\mathrm{S}\,|\,\mathrm{S}}{\mathrm{S}}}{\begin{array}{c}\textit{Someone entered. He left}\\ \exists y.\lambda x.[\,]\,y\end{array}}}{\&(\textbf{entered}\,y)(\textbf{left}\,x)} \;=\; \begin{array}{c}\text{LOWER, beta}\\ \Rightarrow\end{array} \;\begin{array}{c}\mathrm{S}\\ \textit{Someone entered. He left}\\ \exists y.\&(\textbf{entered}\,y)(\textbf{left}\,y)\end{array}$$

In this analysis, the scope of the indefinite determiner extends over the subsequent clause. On the treatment here, this is just the indefinite taking wide scope over more than one clause; see chapter 9 for a more thorough discussion.

The same mechanism that explains crossover above is operative here as well, and predicts that no matter what the scope of the indefinite, it will only be able to bind pronouns that are evaluated after it. Thus a discourse containing the same sentences in reverse order (*He left. Someone entered.*) is correctly predicted not to have an interpretation on which the pronoun covaries with the indefinite.

Note that, unlike some dynamic treatments such as Heim (1983), there is no need here to stipulate any order-sensitive details in the lexical entry of the sequencing operator that conjoins sentences in a discourse. Schlenker (2007), among others, criticizes the dynamic approach for such stipulations. Rather, here, the order asymmetry is part of the general compositional schema, independent of the lexical details of conjunction or any other operator. In fact, the conjunction plays no active role in the anaphoric link established in the derivation above, beyond merely allowing the syntactic part of the link to pass through unimpeded. This is achieved by having the basic lexical entry for conjunction (here, (S\S)/S) undergo the LIFT schema with $A = \mathrm{DP}\rhd\mathrm{S}$. (See section 7.1 for a more general treatment of coordination that accommodates coordination of a wider range of syntactic categories.)

The ability of the continuation grammar to account for dynamic binding and order asymmetries in a principled and general way depends on analyzing clause meaning as continuation-aware, that is, as a function on sentence updates. Analyzing sentences as category $\dfrac{\mathrm{S}\,|\,\mathrm{S}}{\mathrm{S}}$ instead of just category S allows binding information to travel along the upper layer of the diagrams. Because the combination schema introduced in chapter 1 is inherently left-to-right, it provides a unified explanation for order sensitivity in crossover and in anaphora.

3.3 Comparison with Dynamic Montague Grammar

The dynamic approach to natural language meaning includes Kamp (1981), Heim (1982), Groenendijk and Stokhof (1991), Groenendijk and Stokhof (1990), and many

more; see Dekker (2012) for a recent perspective. We concentrate here on Groenendijk and Stokhof's (1990) Dynamic Montague Grammar (DMG), since it is a paradigm example of a dynamic treatment that has some striking similarities to our approach, yet with significant technical, empirical, and philosophical differences.

DMG introduces a type-shifter '↑' to turn a static clause meaning q into its dynamic counterpart $\uparrow q$.

(45) $\uparrow q = \lambda p.\, q \wedge {}^{\vee}p$

In this definition, the down operator ${}^{\vee}$ (a symbol borrowed from intensional logic, but given a different meaning than usual) deals in assignments ('states') rather than worlds.

The connective ';' conjoins two dynamic sentence meanings.

(46) $\phi; \psi = \lambda p.\, \phi({}^{\wedge}\psi(p))$

For example, *John walks and John talks* translates as

(47) $(\uparrow \mathbf{walk}(\mathbf{j}))\,;\,(\uparrow \mathbf{talk}(\mathbf{j})) = \lambda p.\, \mathbf{walk}(\mathbf{j}) \wedge \mathbf{talk}(\mathbf{j}) \wedge {}^{\vee}p.$

An additional type-shifter '↓' extracts a static truth condition from a dynamic sentence meaning.

(48) $\downarrow \phi = \phi({}^{\wedge}\mathbf{true})$

For example, applying ↓ to (47) yields the truth condition

(49) $\mathbf{walk}(\mathbf{j}) \wedge \mathbf{talk}(\mathbf{j}) \wedge {}^{\vee\wedge}\mathbf{true} = \mathbf{walk}(\mathbf{j}) \wedge \mathbf{talk}(\mathbf{j}).$

These elements of DMG manage the composition of a proposition p in (45), (46), and (48) much as the elements of our system manage continuations: roughly, DMG's '↑' corresponds to (a special case of) our LIFT; DMG's ';' corresponds to (a special case of) our combination schema (16) and (likewise) stipulates left-to-right evaluation; and DMG's '↓' corresponds to our LOWER. Indeed, when Groenendijk and Stokhof (1990) write that "we can look upon the propositions which form the extension of a sentence as something giving the truth conditional contents of its possible continuations", they use the word 'continuation' in an informal sense that coincides closely with what it means for us in the context of a discussion of clause meaning.

Is DMG a continuation semantics, then? Perhaps, but if so, an incomplete one. DMG only lifts clause meanings: from $\mathbf{walks}(\mathbf{j})$ to $\uparrow \mathbf{walks}(\mathbf{j})$. Analogously, for quantification, Montague's PTQ only lifts noun-phrase meanings: from the individual $\mathbf{j}$ to the continuation-consumer $\lambda\kappa.\, \kappa\mathbf{j}$. In contrast to DMG and PTQ, our grammar allows lifting any constituent, not just sentences or noun phrases. This uniformity allows our analysis to treat the mechanism by which quantifiers find their scopes as one and the same as the mechanism by which pronouns find their antecedents.

3.4 Unification of generalized quantifiers with dynamic semantics

So Montague continuizes only the category DP. Dynamic semantics continuizes only the category S. The continuation strategy advocated here continuizes uniformly throughout the grammar, including DP and S as special cases of a systematic pattern. As a result, continuations unify the generalized quantifier conception of DP meaning and the dynamic view of sentences as context update functions. They both are instances of the same shift in perspective, on which expressions can not only denote values, but also functions on their own semantic context.