On the semantics of exceptional scope

by

Simon Charlow

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Linguistics

New York University

September, 2014

Prof. Chris Barker

Acknowledgements

I owe a great deal to my committee: Chris Barker (my chair), Lucas Champollion, Philippe Schlenker, Yael Sharvit, Anna Szabolcsi, and Seth Yalcin. Because of their incisive comments, probing questions, and patience, what follows is vastly better than it might have been. In particular, I'd like to mention my gratitude to Lucas for extended conversations and extensive written feedback, to Seth for his keen focus on the big picture, and to Yael for first-hand instruction in the craft of semantics. Philippe offered astute, valuable advice at several critical points. Anna possesses more knowledge about the empirical and theoretical issues at the heart of this work than I can fathom, and she was generous enough to share it with me. Chris gave lavishly of his time, expertise, experience, and example. His influence is on every page.

This dissertation began as an attempt to scratch an itch (why is sloppy anaphora so much less constrained than so many theories of the syntax-semantics interface lead you to expect?) that I've had since my undergraduate honors thesis. For a while, I thought I'd moved on. Then I wrote a qualifying paper on the syntax of ellipsis under the guidance of Mark Baltin (chair), Kyle Johnson, and Anna Szabolcsi—and found out I hadn't. Thanks to my committee for helping me see the light.

Svetlana Godjevac and Ben Russell taught me linguistics as an undergraduate. (Svetlana and Chris are married, and so I got to celebrate my defense with the alpha and omega of my official linguistics education. It meant a lot to me.) Julie Sedivy's semantics tutelage was the bedrock on which everything rests. Polly Jacobson is the reason I'm a linguist; her legendary intro semantics course and uncompromising values left indelible marks. This dissertation wouldn't exist without the seminar on computer science for linguists taught by Chris and Jim Pryor in the fall of 2010.

This work benefited from conversations with colleagues and friends. In addition to the folks already mentioned, thanks to Chris Potts, Paul Portner, Richard Larson, Matthew Barros, Veneeta Dayal, Chung-chieh Shan, Scott AnderBois, Jane Grimshaw, Ivano Ciardelli, Matthijs Westera, Angelika Kratzer, Roger Schwarzschild, Daniel Hardt, Mark Baker, Ken Safir, Jason Merchant, Ezra Keshet, Martin Hackl, Manfred Krifka, Matthew Stone, Jeroen van Craenenbroeck, Uli Sauerland, Jeremy Kuhn, Adina Williams, and Paul Elbourne. The dogged support of Diamandis Gafos and Stephanie Harves deserves special mention, as do Rajesh Bhatt's mixes and kindness. Thanks to audiences and/or reviewers for talks I gave at at SALT 18, The 18th Amsterdam Colloquium, and Rutgers University.

Dylan Bumford gets his own paragraph. A collaborator in everything but name (I hope that changes someday), Dylan was unaccountably good at grasping what I was trying to do with this stuff—even in its uglier and overly complicated embryonic stages—and incredibly indulgent with his time and feedback.

Salvador Mascarenhas was a brilliant fellow traveler for six years. Mike Solomon wrote a qualifying paper that first piqued my interest in alternatives-based semantics for indefinites and tolerated my gchats. Tim Leffel and I shared beer and emoji. Justin Garrison was my roommate when a tornado ripped the roof from our Brooklyn brownstone in 2010 (yes). Emma Cunningham was instrumental early and often. I spent several happy years sharing an office with Inna Livitz and

Jim Wood. Amanda Dye and I understand each other. Donnie Luu's selfless support helped me run several gauntlets.

I wouldn't have surmised I could go to grad school, let alone be an academic, without my brother Nate blazing trails while I bided my time. I hope I do it half as well as him. Thanks, finally, to Mom and Dad. This is for you.

Abstract

I motivate a new theory of exceptional scope phenomena in natural language—that is, the ability of some expressions to affect the interpretation of others from inside scope islands. I propose that a scope island is any domain that is obligatorily evaluated and that exceptional scope uniformly reflects linguistic side effects taking scope after the evaluation of a scope island.

After a high-level overview of the empirical domain and analysis in Chapter 1, Part I develops a new account of linguistic side effects using the category-theoretic notion of a monad and the computer science concept of continuations. Building on dynamic approaches to the semantics of indefinites and anaphora, Chapter 2 motivates a semantics that recognizes two side effects, namely nondeterminism and state modification, and develops a novel approach to dynamic interpretation along these lines using something I call the State.Set monad. Chapter 3 integrates this semantics with a directly compositional treatment of scope-taking which generalizes previous linguistically oriented uses of continuations and crucially allows side effects to take scope.

Part II applies the theory to exceptional scope-taking. Inspired by computer science research on delimited control, Chapter 4 proposes that a scope island is any constituent that gets evaluated in the course of semantic composition. Scope islands, in other words, are semantic phases. It is shown that any side effects which survive evaluation are free to take scope post-evaluation via the same mechanisms which underwrite scope-taking throughout the grammar. The central result is an explanation for why the scope-taking of indefinites appears not to be bounded by islands: simply put, indefinites incur side effects which may take scope after evaluation. Taking this perspective derives a number of empirical properties characteristic of exceptional scope-taking and unaccounted for in existing proposals and suggests novel, unified analyses in a variety of empirical domains. Chapter 5 explores several such cases in greater detail, with specific reference to sloppy anaphora in ellipsis, maximal discourse reference, and association with focus.

The theory turns out to be agnostic in an important respect: whichever side effects a semanticist elects to include in his or her toolkit, the prediction is that they will be the sorts of things that can scope out of islands. I draw this out by considering a semantics which recognizes nondeterministic side effects but offers a less general treatment of anaphora than the State.Set monad. In addition to facilitating a previously elusive compositional formulation of static alternatives-based theories of indefinites, this semantics sheds light on a fundamental connection between alternatives-based and dynamic approaches to indefinites and indefiniteness, while offering a precise accounting of the way in which they differ.

Contents

Chapter 1

Introduction

1.1 Overview

My main claim is that exceptional scope results when linguistic side effects take scope after the evaluation of a scope island. Let me flesh this out a bit.

i. **Exceptional scope** refers to the ability of certain expressions, when embedded in a scope island (e.g. May 1985), to affect the interpretation of expressions outside the island, especially in light of the *in*ability of other expressions to do the same.

ii. **Side effects** are things that happen in the course of composition besides functional application on values (e.g. Barker 2002, Shan 2002, 2005, Shan & Barker 2006, Barker & Shan 2014). To wit, though the Fregean bread and butter of compositional semantics is applying functions to arguments, we often ask our theories to do more, i.e. to handle world- and assignment-sensitivity, nondeterminism, and so on, often via enriched notions of composition.

iii. **Evaluation** is most easily understood (for now) in terms of an LF-based metaphor (we'll eventually settle on a denotational formalization): a constituent is considered evaluated iff nothing has covertly moved (i.e. Quantifier Raised) out of it. Derivatively, side effects **survive** evaluation whenever an evaluated constituent itself gives rise to side effects.

To help get our empirical bearings, here's a sampling of some data I'll analyze in these terms, followed by a sketch of how the analysis goes. I'll use ⟨angled brackets⟩ to identify scope islands. We'll see cases where an indefinite provides an antecedent for (that is, *binds*) pronouns in separate sentences such as (1.1) (e.g. Heim 1982, Kamp 1981), cases where an indefinite projects its existential force out of scope islands such as the reading of (1.2) on which I've got one rich relative whose demise nets me a condo (e.g. Fodor & Sag 1982, Farkas 1981), and cases where an indefinite does both at the same time such as (1.3), which is only grammatical on a reading where there's an expert on indefinites such that every grad hopes for her attendance. (e.g. Abusch 1994, Chierchia 2005).

(1.1) ⟨A linguist$_i$ entered the room⟩. He$_i$ sat on the floor.

(1.2) If ⟨a wealthy relative of mine dies⟩, I'll inherit a condo.

(1.3) Every grad had hoped ⟨a renowned expert on indefinites$_i$ would be at the conference⟩, but her$_i$ flight was cancelled at the last minute.

In addition, we will analyze cases of ellipsis involving *sloppy anaphora* (e.g. Keenan 1971, Sag 1976, Tomioka 1999) to a deeply embedded antecedent such as (1.4), cases with anaphora to a *maximal discourse referent* (e.g. Kamp & Reyle 1993, van den Berg 1996, Nouwen 2007, Brasoveanu 2007) created by a deeply embedded quantifier such as (1.5), and cases such as (1.6) where a focused item associates with a focus-sensitive adverb, despite the fact that a scope island intervenes between the two (e.g. Rooth 1985, Kratzer 1991, Krifka 1991, 2006, Wold 1996).

(1.4) If everyone thinks John$_i$ will come, we'll have to invite him$_i$.
If everyone thinks ⟨BILL$_j$ will⟩, we WON'T ~~have to invite him$_j$~~.

(1.5) Everyone heard the rumor that ⟨at least six [senators]$_i$ [supported Cruz's filibuster]$_j$⟩. It turned out to be erroneous: they$_{i \cap j}$ numbered at most three.

(1.6) I $\underbrace{\text{only said I'd be upset if } \langle\text{BIL}}L_\text{F}$ came⟩.

I claim these phenomena have a unified, semantic explanation: the sorts of side effects an expression incurs determines whether it takes exceptional scope—and *how*. Thus, (1.2) reflects *nondeterministic side effects* taking scope after evaluation of the bracketed scope island, (1.4) and (1.5) reflect *state-changing* side effects taking scope after evaluation, (1.1) and (1.3) reflect *both* nondeterministic *and* state-changing side effects taking scope after evaluation, and (1.6) reflects *focus side effects* (i.e. alternatives) taking scope after evaluation. Underlying each of these analyses is a simple, minimal shift in perspective, to recasting the insights of dynamic semantics (e.g. Groenendijk & Stokhof 1991, Dekker 1994, Muskens 1996, Brasoveanu 2007) and alternative semantics (e.g. Rooth 1985, 1992b) *in terms of side effects*. In other words, what's required to analyze these data (and more besides) is in a sense implicit in extant analyses. Even so, I'll argue that the shift to a side-effects-centric perspective has conceptual and especially empirical virtues relative to extant theories (see Section 1.3 for a preview), and so it is a shift worth making.

1.2 Theory: side effects and scope

Part I introduces theoretical tools for talking about side effects, scope, and evaluation. The technical pieces I rely on are a theory of side effects (in terms of *monads*), a theory of scope-taking (in terms of *scopal composition* via *continuations*), and a way to link the two (in terms of scopal composition in the presence of an *underlying monad*). I show how to formulate a version of standard dynamic semantics for indefinites and cross-sentential anaphora in terms of side effects (specifically, *nondeterminism* and *state*) and integrate it into the broader theoretical landscape. This sets the stage for the advances reported in Part II, where I show that the theory developed in Part I allows nondeterministic and state-changing side effects to take scope out of islands.

Side effects and monads. Side effects are things that happen in the course of composition besides functional application on values. Though this sounds exotic, side effects are ubiquitous in semantic theorizing. A familiar example (emphasized by Shan 2002, 2005) is the primary mode of combination in the textbook semantics of Heim & Kratzer 1998: to accommodate objects whose meanings can shift depending on the *state* (i.e. index) they're evaluated at (for Heim & Kratzer, pronouns and constituents containing unbound pronouns), Heim & Kratzer suppose that the grammar is in the business of composing meanings which are *state-sensitive* (a variety of data structures can play the role of states; Heim & Kratzer use assignment functions). Another common example is the idea that some meanings have *nondeterministic* side effects—i.e. they prompt us to consider a number of computations in parallel. Nondeterministic side effects motivate a mode of composition that nondeterministically applies the values in one set to the values in another, i.e. extends functional application into point-wise functional application on sets of values (e.g. Hamblin 1973). A final example handles meanings with both state-sensitive and nondeterministic side effects—meanings that return different nondeterministic values depending on the state they're evaluated at. The corresponding enriched mode of composition is state-sensitive nondeterministic functional application (e.g. Kratzer & Shimoyama 2002, Alonso-Ovalle 2006, Shimoyama 2006). Here is a summary of these remarks, with a bit of formal detail:

Operation	Semantics
Functional application	$m\,n$
State-sensitive functional application	$\lambda s.\, m\, s\, (n\, s)$
Nondeterministic application	$\{x\, y \mid x \in m \wedge y \in n\}$
State-sensitive nondeterministic functional application	$\lambda s.\, \{x\, y \mid x \in m\, s \wedge y \in n\, s\}$

Following Shan's 2002 pioneering work, I use *monads* (e.g. Moggi 1989, Wadler 1992, 1994, 1995) as a way to think a bit more abstractly and systematically about what we do when we posit grammars that allow expressions to incur side effects. In other words, monads give a *theory of side effects*. Again, this sounds exotic, but a monad is nothing more than a construct for modeling a given side effects regime in terms of the λ-calculus (real-life examples coming shortly). A given monad $\mathcal{M}$ does two things:

i. Characterizes what *programs* in $\mathcal{M}$ (i.e. things that *may* incur side effects) are, and what distinguishes *impure* programs in $\mathcal{M}$ (programs that *do* incur side effects) from *pure* programs (those that *don't*).

ii. Determines a notion of program *sequencing*—that is, of how complex programs in $\mathcal{M}$ can be built up from smaller programs in $\mathcal{M}$.

More formally, a monad is a triple $\langle \mathsf{M}, \eta, \multimap \rangle$. M is a *type constructor* relating the types of values with the types of programs. A monad determines a pair of functions $\langle \eta, \multimap \rangle$. The first of these, η (type $\alpha \to \mathsf{M}\alpha$), is an *injection* function that upgrades a value a into a trivial/pure program a^{η}, i.e. one without side effects (conversely, impure programs incur side effects and cannot be derived by applying η to any value); it thus tells us what sorts of programs inhabit $\mathcal{M}$, while distinguishing pure programs (those that can be derived from a value via η) from impure programs (those that cannot). The second function, $(\multimap)$ (type $\mathsf{M}\alpha \to (\alpha \to \mathsf{M}\beta) \to \mathsf{M}\beta$), is a recipe for *sequencing* programs in the service of compositionally assembling complex programs from smaller pieces—in other words, precisely what we need in our toolkit if we're using monads to compositionally assemble linguistic programs with side effects. (In order for a given $\mathcal{M}$ to qualify as a monad, its η and $(\multimap)$ must additionally satisfy a trio of properties known as the "Monad Laws".)

Compositional regimes familiar to semanticists can be equivalently formulated in monadic terms. For example, functional application on values (i.e. composition sans side effects) is captured by the *Identity* monad, state-sensitive application by the *Reader* monad, nondeterministic application by the *Set* monad, and state-sensitive nondeterministic application by the composite *Reader.Set* monad:[1]

Side effects	Monad	$\mathsf{M}\alpha$	a^η	$m \multimap k$
None	Identity	α	a	$k\,m$
State-sensitivity	Reader	$s \to \alpha$	$\lambda s.\, a$	$\lambda s.\, k\,(m\,s)\,s$
Nondeterminism	Set	$\alpha \to t$	$\{a\}$	$\bigcup_{a \in m} k\,a$
State-sensitivity, nondeterminism	Reader.Set	$s \to \alpha \to t$	$\lambda s.\, \{a\}$	$\lambda s.\, \bigcup_{a \in m\,s} k\,a\,s$

These monads are in direct correspondence with the modes of composition discussed above. Thus, monads are a *perspective*, not a radical (or substantive) claim about the sorts of meanings we need in our toolbox. Nevertheless, I argue, taking this perspective advances the theory of exceptional scope.

Dynamic side effects. Throughout the dissertation, I'm guided by a commitment to characterizing phenomena motivating extended varieties of semantics in terms of side effects (see also Shan 2002):

> Monadic Credo (after Lewis 1970): to say what meaning *is*, we may first ask what meanings *can do*, and then find a monad that lets them do that.

Along these lines, I propose to refactor *dynamic semantics for anaphora* in terms of linguistic side effects. This is more than an exercise; the resulting semantics is the bedrock on which the analyses to come rest. I begin with a familiar datum. Indefinites like *a linguist* readily bind pronouns in subsequent sentences, even as binding of this sort is impossible for quantifiers like *no linguist*:

(1.7) ⟨{A,*No} linguist$_i$ entered the room⟩. He$_i$ sat on the floor.

(*exceptional anaphora to an indefinte*)

An influential response to data like this has been to develop *dynamic* theories of meaning which treat (some) meanings as instructions for updating the discourse context, and pronouns as things which look to the discourse context for their meaning (e.g. Karttunen 1976, Heim 1982, Kamp 1981,

1 The Reader monad is also known as the Environment monad. These names are part of the functional programming lore and vernacular; their ultimate source is not known to me.

Barwise 1987, Rooth 1987, Groenendijk & Stokhof 1991, Dekker 1994, van den Berg 1996, Muskens 1996, Bittner 2001, Brasoveanu 2007, 2008). Standard dynamic semantics has two essential pieces:

i. **Changing state.** Dynamic semantics supposes that part of what interpretation is about is *creating discourse referents* (drefs), and models this in terms of meanings which take as an input (some feature of) the conversational context, modify the input by adding to it some information about the objects under discussion, and *output* the modified input.

ii. **Nondeterminism.** Dynamic semantics models indefinites as the nondeterministic analogs of proper names, thereby allowing indefinites to *nondeterministically* add a dref (e.g. for *a linguist* some nondeterministic linguist) to the modified output.

In other words, things can *modify* the "conversational scoreboard" by introducing drefs, and indefinites can do so *nondeterministically*. In the same way an utterance of *Bill entered the room* can make Bill available qua dref for anaphoric reference, an utterance of *a linguist entered the room* can make some linguist or other available qua nondeterministic dref.

I cast a computational eye towards dynamic semantics, constructing a grammar that treats the ability to create drefs and the ability to do so nondeterministically *as side effects*. In other words, I go on the hunt for a monad inhabited by programs that may incur state-changing and nondeterministic side effects. In fact, we're already halfway there. The Set monad is, as I've noted, a natural choice for modeling nondeterministic side effects. What remains is to find a monad for state-changing effects—i.e. a monad that allows updates of the "scoreboard". Such a monad exists and (fittingly) is known as the *State* monad. A proper generalization of the Reader monad, the State monad allows for programs that take inputs, possibly modify them, and then, finally and crucially, *output* the possibly-modified inputs (see Giorgolo & Unger 2009, Unger 2012 for uses of State monads in analyses of discourse anaphora distinct from my own and Giorgolo & Asudeh 2012 for an analysis of conventional implicatures in terms of a State monad).

Once we alight on the State and Set monads, the work of assembling a monad for dynamic semantics is essentially complete. The shape of a combined monad for both state-changing and nondeterministic side effects turns out to be fully determined. I call the result the *State.Set* monad:

Definition 1.1 (The State.Set monad).

$$\begin{aligned} \mathsf{M}\alpha &::= s \to (\alpha \times s) \to t \\ a^{\eta} &:= \lambda s.\, \{\langle a,\ s\rangle\} \\ m \multimap k &:= \lambda s. \bigcup_{\langle a, s'\rangle \in m\, s} k\, a\, s' \end{aligned}$$

The State.Set monad underwrites a new type of dynamic semantics—albeit one closely related to the classic varieties cited above. (Skip ahead a couple pages to Definition 1.3 to see examples of meanings that exploit the State.Set monad.) This semantics treats state and nondeterminism as side effects, the basis for the empirical advances detailed in Chapters 4 and 5 and previewed in Section 1.3.

Scope. Chapter 3 integrates side effects with scope-taking. I introduce a recent approach to scopal composition advocated by Barker 2002, Shan & Barker 2006, Barker & Shan 2008, 2014, then pursue a linguistically novel integration of scopal composition and side effects in terms of something I call an *underlying monad*.

Barker 2002, Shan & Barker 2006, de Groote 2006, Barker & Shan 2008, 2014 argue that (e.g. quantificational) scope-taking in natural language is fruitfully analyzed by appealing to a *scopal* notion of composition. Grammars that deal with scope-taking by extending the semantics of composition in this way are known as *continuations-based* approaches to composition. In continuations-based grammars, everything potentially denotes a scope-taker, and two scope-takers combine by an operation of *scopal application*. Scopal application ('**S**') is in a sense a semantic way of constructing an LF. It provides scope-takers with their scopes as a matter of interpretation in the service of compositionally assembling complex scope-takers from smaller pieces:

$$\mathsf{S}\, m\, n \coloneqq \lambda k.\, m\, (\lambda x.\, n\, (\lambda y.\, k\, (x\, y)))$$

Because scopal application requires m and n to be scopal in order to work its magic, and because not everything starts off as scopal (e.g. proper names, transitive verbs, etc.), continuations-based composition requires a way to *Lift* expressions into scope-takers ('$\uparrow$'). In standard continuations-based theories (ibid.), the result of Lifting an expression a into a scope-taker is just the "Montague-lift" of a, namely $\lambda k.\, k\, a$ (e.g. Montague 1974, Partee 1986). In addition, continuations-based grammars require a *Lower* operation ('$\downarrow$'), a way to change an expression that takes scope into one that does not. In standard continuations-based theories, Lower is accomplished by applying a scopal program to a *trivial* continuation/scope k_0, namely the identity function $\lambda a.\, a$.

Linking side effects and scope. I argue that extant continuations-based semantics represent just one instantiation of a more general perspective, one that implicates monads in an essential way. The central insight is that the Lift and Lower of Barker 2002, Shan & Barker 2006, Barker & Shan 2008, 2014 can be generalized to *monadic* Lift and Lower. More specifically, given a monad $\mathcal{M}$, Lift can be identified with $\mathcal{M}$'s sequencing operation ($\multimap$) (i.e. the second argument of ($\multimap$) can be construed as a continuation/scope), and the trivial continuation/scope k_0 on which Lower is built can be identified with $\mathcal{M}$'s injection function η (the general approach is inspired by Liang et al.'s 1995 ContT transformer and Wadler's 1994 approach to inner monads and delimited continuations):

Definition 1.2. A *grammar* is a pair $\langle \mathsf{S}, \mathcal{M}\rangle$ of scopal application (**S**) and an underlying monad $\langle \mathsf{M}, \eta, \multimap\rangle$ such that $\uparrow \coloneqq (\multimap)$ and $k_0 \coloneqq \eta$.

Theory	$m^{\uparrow}$	k_0
Previous work	$\lambda k.\, k\, m$	$\lambda a.\, a$
My proposal	$\lambda k.\, m \multimap k$	η

This approach builds a bridge between side effects and scope-taking. In other words, it enables side effects to *take scope*, including (crucially) *after evaluation*.

Thus, analogous to how monads can be used to extend functional application into operations on programs, scopal composition can be extended into a recipe for scopally assembling programs given different underlying monads. A semantics where $m^{\uparrow} := \lambda k.\, k\, m$ and $k_0 := \lambda a.\, a$ represents just one instantiation of this general pattern, namely the one that results when identify $\mathcal{M}$ with the *Identity* monad—i.e. the monad inhabited by trivial programs. Another, equally valid option (and one motivated by the discussion in Chapter 2) is to fix the underlying monad to the State.Set monad, that is, to pair scopal application with state-changing and nondeterministic side effects.

I fix an underlying State.Set monad and some lexical entries. See Definition 1.3 for a small sampling (‘$m_v \multimap \pi$’ abbreviates ‘$m \multimap \lambda v.\, \pi$’). Notice that only those meanings that incur side effects need to be defined in terms of the underlying monad; the rest keep their familiar denotations. (This is characteristic of the modularity delivered by monads as a programming technique, see e.g. Wadler 1995, Shan 2002.) Next, I show how to scopally compose and evaluate various cases of interest, including surface scope, binding, inverse scope, and the composition of relative clauses, complex nominals, and complex DPs. The resulting semantics thus has a reasonable degree of coverage with respect to “core” linguistic phenomena. Armed with it, we move on to Part II and the account of exceptional scope.

Definition 1.3 (State.Set “lexicon” sampling).

Item	**Meaning**	**Type**
John	j	e
saw	saw	$e \to e \to t$
and	$(\wedge)$	$t \to t \to t$
a linguist	$\textbf{a.ling} := \lambda s.\, \{\langle x,\, s\rangle \mid \mathsf{ling}\, x\}$	$\mathsf{M}e$
not	$\textbf{not} := \lambda m s.\, \{\langle \neg\exists s'.\, \langle \mathsf{True},\, s'\rangle \in m\, s,\, s\rangle\}$	$\mathsf{M}t \to \mathsf{M}t$
if	$\textbf{if} := \lambda m n.\, \textbf{not}\, (m_p \multimap (\textbf{not}\, n)_q \multimap (p \wedge q)^{\eta})$	$\mathsf{M}t \to \mathsf{M}t \to \mathsf{M}t$
every linguist	$\textbf{ev.ling} := \lambda k.\, \textbf{not}\, ((\textbf{a.ling})_x \multimap \textbf{not}\, (k\, x))$	$(e \to \mathsf{M}t) \to \mathsf{M}t$

1.3 Application: scope-taking after evaluation

Part II applies this semantics—scopal application in the presence of an underlying monad—to empirical issues. I show that a seemingly disparate group of data can be profitably and uniformly analyzed in terms of one simple fact about the grammar: scope islands are constituents that must be evaluated, and any side effects which survive evaluation are free to take scope post-evaluation.

Islands and exceptionally scoping indefiniteness. The central empirical payoff, in Chapter 4, is an analysis of indefiniteness qua nondeterminism that explains exceptional scope-taking, connects it with nondeterministic discourse anaphora, and achieves greater empirical coverage than extant treatments of indefiniteness (in both dynamic and static traditions). Let’s review a datum I mentioned at the outset. In addition to binding out of scope islands, indefinites project their quantificational force scope out of scope islands:

(1.8) If ⟨a wealthy relative of mine dies⟩, I'll inherit a condo. (*exceptional existential scope*)

Inspired by computer science research on *delimited control* (e.g. Danvy & Filinski 1990, Wadler 1994), I propose that a scope island is any domain that is *obligatorily evaluated*—equivalently, any domain which is obligatorily Lowered (the idea bears an intriguing resemblance to Chomsky's 2008 notion of phases as the computational domains of syntactic theory):

Definition 1.4. *Scope islands* are constituents that have to be evaluated. An *evaluated* constituent is any constituent that has been Lowered, i.e. combined with a trivial continuation.

Above, I characterized evaluation in operational terms, such that a constituent is considered evaluated iff nothing has QR'd out of it. The proposal mooted now is essentially a semanticization of the operational way of stating things: evaluating a constituent means delimiting the scope-taking of any and all scopal expressions inside it; an evaluated constituent is one whose every scopal expression has been united with its continuation—i.e. taken scope.

Though the grammar insists scope islands be evaluated, in many cases side effects *survive evaluation*. The upshot is a *semantic explanation* for why indefinites can impose both anaphoric and nondeterministic effects on things beyond the boundaries of a scope island. Because $\uparrow$ is identified with $\multimap$, any side effects that survive evaluation are free to take scope post-evaluation: sequencing an evaluated, impure program π with a new continuation/scope argument k means that any side effects in π inevitably influence the evaluation of k.[2]

See Facts 1.1 and 1.2 below for a couple examples. The "tower notation" used here, which stacks meaning elements on top of each other, is at this point only intended to be evocative: roughly, the top level of a tower is a scopal tier where side effects live, and the bottom level is a scope-less tier where values live (more concretely, the right-most towers in Facts 1.1 and 1.2 are just alternative representations of the scopal functions $\lambda k.\, \textbf{a.ling}^{\triangleright}_{x} \multimap k\, (\textsf{left}\, x)$ and $\lambda k.\, k\, (\forall x.\, \textsf{ling}\, x \Rightarrow \textsf{left}\, x)$). In both cases, the scope island is evaluated via an application of $\downarrow$ and re-Lifted into a scope-taker via an application of $\uparrow$, completing what's known in the computer science literature as a *Reset*. In Fact 1.1, the input to $\uparrow$ (i.e. the underbraced bit) harbors both nondeterministic and state-changing side effects, which climb back onto the scopal tier post-Reset (nondeterministic effects are incurred by **a.ling**, and state-changing side effects by the introduction operator $\triangleright$). In Fact 1.2, the input to $\uparrow$ is pure, and nothing makes it back up to the scopal tier post-Reset.

Fact 1.1 (Resetting *a linguist left*).

$$\left(\frac{\textbf{a.ling}^{\triangleright}_{x} \multimap [\ ]}{\textsf{left}\, x}\right)^{\downarrow\uparrow} = (\underbrace{\textbf{a.ling}^{\triangleright}_{x} \multimap (\textsf{left}\, x)^{\eta}}_{\text{State- and Set- impure}})^{\uparrow} = \frac{\textbf{a.ling}^{\triangleright}_{x} \multimap [\ ]}{\textsf{left}\, x}$$

2 There is an intriguing connection between my proposal and theories such as Dayal 1996, 2002, Krifka 2006: even as an expression's scope-taking is bounded by islands, its effects can be felt beyond the border of the island.

Fact 1.2 (Resetting *every linguist left*).

$$\left(\frac{\textsf{\textbf{ev.ling}}\,(\lambda x.\,[\,])}{\textsf{left}\,x}\right)^{\downarrow\uparrow} = (\underbrace{(\forall x.\,\textsf{ling}\,x \Rightarrow \textsf{left}\,x)^{\eta}}_{\text{pure}})^{\uparrow} = \frac{[\,]}{\forall x.\,\textsf{ling}\,x \Rightarrow \textsf{left}\,x}$$

Thus, given an underlying State.Set monad, we predict that both state-changing and nondeterministic side effects will be subject to exceptional scope-taking, as in (1.1), (1.2), and (1.3). Both sorts of side effects will survive evaluation and are free to take scope post-evaluation via the same mechanisms which undergird scopal composition more generally in the grammar. By contrast, expressions such as *every linguist*, which fail to trigger state-changing and/or nondeterministic side effects (and wipe out certain side effects generated within their scope), fail to take exceptional scope (and exhaust the scope-taking abilities of those side effects incurred within their scope).

Other core data are accounted for along similar lines. Because nothing prevents a side effect that survives evaluation from ending up in the scope of some structurally higher operator, we explain why indefinites can take intermediate exceptional scope, i.e. why (1.9) can mean that for each student x there is some Chomskyan condition y such that x has to come up with three arguments against y. In addition, because distributivity operators are essentially universal quantifiers, we predict that their scope will, like other universal quantifiers, be bounded by scope islands. This explains why (1.10) can mean that two of my relatives are such that *if they each die*, I'll inherit a condo, but not that two of my relatives *are each such that if they die*, I'll inherit a condo.

(1.9) Each student has to come up with three arguments which show that ⟨some condition proposed by Chomsky is wrong⟩.
(Farkas 1981, ex. 17a) (*intermediate exceptional scope*)

(1.10) If ⟨two relatives of mine die⟩, I'll inherit a condo.
(after Ruys 1992: 107–108) (*island-bounded distributivity*)

The way side effects acquire scope post-evaluation here is fundamentally no different from the way they acquire scope at any other time. This has a couple important consequences. First, we explain why exceptional scope-taking *feeds* anaphoric processes in cases such as (1.3), repeated below as (1.11): when an indefinite's side effects take exceptional scope over an operator, any discourse referents induced by the indefinite inevitably get brought along for the ride. Second, we explain why a restricted indefinite's side effects cannot take scope above any operator that binds into them—for example, why (1.12) lacks a reading where the indefinite out-scopes *no candidate*—something dubbed the *Binder Roof Constraint* by Brasoveanu & Farkas 2011: since there is but one notion of scope, the unattested reading is simply semantically incoherent.

(1.11) Every grad had hoped ⟨a renowned expert on indefinites$_i$ would be at the conference⟩, but her$_i$ flight was cancelled at the last minute. (*exceptional scope feeds binding*)

(1.12) No candidate$_i$ submitted a paper he$_i$ wrote.
(Schwarz 2001, ex. 25) (*Binder Roof Constraint*)

Both data points are problematic for theories which leave indefinites in situ and rely on "pseudo-scope" mechanisms to derive exceptional scope (e.g. Reinhart 1997, Winter 1997, Kratzer & Shimoyama 2002, Schwarzschild 2002, Brasoveanu & Farkas 2011). Since pseudo-scope theories interpret exceptionally scoping indefinites inside the nearest scope island, they don't explain how the anaphoric accessibility of the indefinite in (1.11) could possibly be determined by where its existential force is felt. Similarly, if indefinites can "take scope" while remaining in situ, nothing should delimit this upward mobility in cases like (1.12) (Schwarz 2001).

The theory I propose additionally predicts the potential *selectivity* of exceptional scope—that is, the ability of multiple indefinites inside a scope island to take scope in different ways beyond the boundaries of the island, as on the one-wealthy-relative, any-persuasive-lawyer reading of (1.13).

(1.13) If ⟨a persuasive lawyer visits a relative of mine⟩, I'll inherit a fortune.

(*selective exceptional existential scope*)

My grammar generates fully evaluated programs with higher-order structure, as for example in Fact 1.3. This meaning (type MM*t*) retains enough structure to allow the two sources of nondeterminism (**a.rel** and **a.law**) to be distinguished post-evaluation. Thus, the meaning in Fact 1.3, though fully evaluated, is fine-grained enough to allow the two chunks of indefiniteness to take scope in different ways beyond the boundaries of the scope island. The other layering, with **a.law** out-scoping **a.rel**, is also possible, as is a flatter meaning of type M*t* with a single undifferentiated mass of nondeterminism.

Fact 1.3 (A higher-order, fully evaluated State.Set meaning).

$$\textbf{a.rel}_y \multimap (\textbf{a.law}_x \multimap (\text{visits}\; y\; x)^\eta)^\eta :: \mathsf{MM}t$$

Higher-order structures of this sort receive independent motivation. For one, higher-order MM*t* programs are analogous to meanings postulated in the closely related domain of *higher-order questions* (e.g. Dayal 1996, 2002, Fox 2012). For another, AnderBois 2010 claims (pace e.g. Chung et al. 1995, Merchant 2001, Barker 2013) that sluicing is grammatical iff the meaning of the sluiced clause is isomorphic to the meaning of the antecedent clause. If AnderBois's arguments are correct, and if a sentence with two indefinites is ambiguous in the way I'm suggesting, the prediction is that a sluiced clause will be able to target one indefinite or the other. This prediction is confirmed. Sentence (1.14) can entail a refusal to identify either the persuasive lawyer or the relative. Languages such as Hungarian with multiple sluicing (and richer cases systems) bear out the full three-way paradigm.

(1.14) A persuasive lawyer visited a relative of mine, but I'll never say who.

(*ambiguous sluicing with multiple indefinites*)

(1.15)

Valaki	megvert	valakit	de	nem	tudom	hogy	{ki, kit, ki kit}.
someone	up-beat	someone-Acc	but	not	know-1sg	SUBORD	{who, whom, who whom}

'Someone beat someone up, but I don't know {who, whom, who whom}.'

(*three possibilities for sluicing in Hungarian*)

These predictions are characteristic of some, but not all, theories of indefiniteness and exceptional scope. For example, Kratzer & Shimoyama's 2002 nondeterminism-based account of exceptional scope forces two indefinites on a scope island to take scope in the same way beyond the boundaries of the scope island,[3] and alternatives-based accounts of sluicing do not generate enough interpretations for sentences with two indefinites and thereby under-generate for cases like (1.14).

Finally, an independently motivated nondeterministic semantics for disjunction (e.g. Groenendijk & Stokhof 1991, Stone 1992, Zimmermann 2000, Alonso-Ovalle 2006, Aloni 2007, Mascarenhas 2009, Groenendijk & Roelofsen 2009) explains why disjunctions can take scope out of scope islands, as on the reading of (1.16) where I'm unsure whether Bill thinks "I hope someone will hire a maid" or "I hope someone will hire a cook". If disjunctions give rise to nondeterministic side effects, it's automatic that these effects can take scope post-evaluation. Moreover, as with indefinites, the possibility of higher-order programs predicts that multiple disjunctions inside a single island can take selective scope outside the island, along with attendant ambiguities in sluicing constructions. I argue that these predictions are confirmed.

(1.16) Bill hopes that ⟨someone will hire a maid or a cook⟩.
(Rooth & Partee 1982, ex. 21c) (*exceptionally scoping disjunction*)

One final note on disjunction. As Rooth & Partee 1982 note, the exceptional-scope reading I just flagged has a curious property: the disjunction scopes over *hopes*, even as the individual disjuncts scope *under* it. Though this looks disturbingly like a violation of the Binder Roof Constraint, I show that it follows from disjunction's inherent polymorphism.

Summing up, I argue that an expression's semantics explains whether and how that expression takes exceptional scope. If something gives rise to nondeterministic and/or state-changing side effects, those side effects can take scope out of scope islands. To the extent that my arguments succeed, they support a dynamic approach to exceptional scope (pace Reinhart 1997). Indeed, the account is compatible with a wide range of dynamic theories (e.g. the recent developments reported in Brasoveanu 2007, 2008). All it requires from a dynamic semantics is a nondeterministic treatment of indefiniteness.

Generalized exceptional scope. My theory predicts exceptional scope-taking to be quite a bit more widespread than has been suspected. *Any and all* side effects are predicted to survive evaluation and take exceptional scope. Chapter 5 argues that this prediction is a good one. In particular, it serves as a basis for new insights into a diverse array of (what I argue to be) exceptional scope phenomena, from domains as varied as ellipsis, maximal discourse reference, and association with focus.

First, I look at *"surprising" sloppy readings* in ellipsis, cases such as (1.17) where there's reason to suppose that the proper name $BILL_j$ binds the elided pronoun him_j from a deeply embedded position. The argument is straightforward: (i) it's widely assumed that binding is a prerequisite for *sloppy* interpretations such as these (i.e. interpretations on which the interpretation of the elided VP seems to differ from that of its antecedent; see e.g. Keenan 1971, Sag 1976, Williams 1977, Rooth

3 Though their theory is specifically targeted at Japanese and German indeterminate pronouns, Kratzer & Shimoyama express optimism that it could extend to indefinites cross-linguistically, including in English-like languages.

1992a, Tomioka 1999, Takahashi & Fox 2005); (ii) here *BILL*$_j$ is separated by a scope island from the nearest position from which it could (dynamically) bind the elided sloppy pronoun *him*$_j$; ergo (iii) exceptional scope of *BILL*$_j$ must be feeding binding of the sloppy pronoun.

(1.17) If everyone thinks John$_i$ will come, we'll have to invite him$_i$.
If everyone thinks ⟨BILL$_j$ will⟩, we WON'T ~~have to invite him$_j$~~.
(*sloppy anaphora to a deeply embedded antecedent*)

Though the possibility of such a binding relationship is unexpected both for dynamic accounts of anaphora and leading theories of exceptionally scoping indefinites, it falls out of the account of exceptional scope-taking proposed here. The account predicts that *any* side effects that survive the obligatory evaluation that happens on the shores of a scope island are free to take exceptional scope post-evaluation in the same way as the side effects incurred by indefinites and disjunctions. Drefs deterministically induced by proper names survive evaluation just as well as nondeterministic drefs and thus are correctly predicted to scope out of islands. As illustrated below, Resetting *Bill comes* has no effect on its anaphoric potential. A state-changing side effect, the introduction of a dref for Bill (via ▷), survives evaluation and is thus free to take scope post-evaluation.

Fact 1.4 (Resetting *BILL comes*).

$$\left(\frac{\mathsf{b}_x^{\eta\triangleright}\multimap [\ ]}{\mathsf{comes}\ x}\right)^{\downarrow\uparrow} = \underbrace{(\mathsf{b}_x^{\eta\triangleright}\multimap (\mathsf{comes}\ x)^{\eta})}_{\text{State-impure}}{}^{\uparrow} = \frac{\mathsf{b}_x^{\eta\triangleright}\multimap [\ ]}{\mathsf{comes}\ x}$$

Though this sort of exceptional scope-taking is often invisible, surprising sloppy readings reveal its presence. In addition, I give evidence that this sort of "invisible" exceptional scope-taking cuts cross-categorially, i.e. that it likewise characterizes the drefs introduced by arbitrary XPs.

Second, the theory of exceptional scope makes good predictions for cases of maximal discourse reference (e.g. Kamp & Reyle 1993, van den Berg 1996, Brasoveanu 2007) to deeply embedded expressions in cases such as (1.18) and (1.19). In the first case, *they* is readily interpreted as roughly synonymous with *the first years*, and in the second, *they* is readily interpreted as roughly synonymous with *the senators who supported Cruz's filibuster*. In both cases, anaphora is possible even as a scope island intervenes between the position where the relevant dref is generated and the nearest position from which the relevant dref is accessible to the plural pronoun *they*.

(1.18) No one could really believe the rumor that ⟨more than three [first-years]$_i$ got an A in phonology⟩. They$_i$'re all semanticists.

(1.19) Everyone heard the rumor that ⟨at least six [senators]$_i$ [supported Cruz's filibuster]$_j$⟩. It turned out they$_{i \cap j}$ numbered at most three.
(*maximal discourse reference to deeply embedded antecedents*)

I focus on (1.19) in this introduction. Very roughly, modern dynamic theories suppose that part of what quantificational expressions like *at least six senators* do is create a maximal dref bottling up a plural individual comprised of all and only the individuals who satisfy the quantifier's restriction and nuclear scope (see e.g. van den Berg 1996, Nouwen 2003, 2007, Brasoveanu 2007, 2008). In (1.19), this dref is comprised of the senators who admire Cruz. However, this maximal dref is generated inside an island, and moreover in a position from which even dynamic theories of anaphora predict it should be inaccessible to the pronoun in the subsequent sentence.

As with surprising sloppy readings, this suggests that maximal drefs have exceptional scope properties. And as with surprising sloppy readings, my proposal for exceptional scope makes the correct prediction. The maximal dref comprised of the senators who admire Cruz is a state-changing side effect. Therefore, it's the sort of thing that survives evaluation, and accordingly the sort of thing that can take continue taking scope post-evaluation. See Fact 1.5 for a depiction of how this goes. Notice that a maximal dref taking exceptional scope doesn't mean the quantifier that generates it takes exceptional scope *qua quantifier*. The quantifier's quantificational force is discharged on evaluation, even as the maximal dref it generates survives.

Fact 1.5 (Resetting *at least six senators supported Cruz*).

$$\left(\frac{\geq\!\mathbf{6.sens}\,(\lambda x.\,[\,])}{\mathsf{supp}\,\mathsf{c}\,x}\right)^{\downarrow\uparrow} = (\underbrace{(\mathsf{sen}\cap\mathsf{supp}\,\mathsf{c})_x^{\eta\triangleright}\!\multimap\,(|x|\geq 6)^{\eta}}_{\text{State-impure}})^{\uparrow} = \frac{(\mathsf{sen}\cap\mathsf{supp}\,\mathsf{c})_x^{\eta\triangleright}\!\multimap\,[\,]}{|x|\geq 6}$$

In other words, cases such as these argue that quantifiers like *at least six senators* in fact *do* give rise to a sort of exceptional scope, contrary to what is generally supposed. The way in which this exceptional scope-taking happens is, however, closer to the exceptional scope properties of proper names than to the nondeterministic exceptional scope-taking of indefinites and disjunction. Its effects can only be detected via anaphoric processes.

The picture that emerges from surprising sloppy readings and exceptional maximal discourse reference is that binding relationships are far less constrained by the syntax than is usually supposed (even by dynamic semanticists). Specifically, the data suggests—and my theory predicts—that anaphoric side effects are *always accessible in principle*, just as nondeterministic side effects are always accessible in principle. This achieves, I submit, something closer to the full promise of dynamic semantics than standard dynamic theories offer: on my account, binding and nondeterministic scope are both truly liberated from syntax. A related perspective on the interaction of syntax and binding has been reached independently in work by Safir 2004 (see also Fiengo & May 1994). Safir's conclusions are consistent with—indeed, derived by—an empirically robust theory of interpretation.

Finally, I give a treatment of association with focus that correctly predicts its insensitivity to islands along with the potential for selective association with focus. As I noted at the outset of this chapter, focus-sensitive operators readily associate with focused expressions across island boundaries:

(1.20) I only said I'd be upset if ⟨BILL$_{\text{F}}$ came⟩. (*association with focus across islands*)

To account for data of this sort, Rooth 1985, 1992b, 1996 proposes that the semantic effect of focus is to invoke alternatives which are managed via an enriched, *two-dimensional* approach to meaning and composition. The basic idea is that the interpretation function $[\![\cdot]\!]$ is replaced with a pair of interpretation functions $\langle [\![\cdot]\!], [\![\cdot]\!]_f \rangle$. The first dimension, $[\![\cdot]\!]$, composes values via vanilla functional application. Focused expressions invoke alternatives in the second dimension: $[\![X_{\text{F}}]\!]_f := \mathsf{Alt}\, [\![X]\!]$, with Alt a function from values a into a set of (contextually relevant) alternatives to a. These alternatives compose with their surroundings via nondeterministic functional application. Nondeterministic application works as a pseudo-scope mechanism, propagating alternatives out of scope islands without relying on island-violating scoping mechanisms.

Using nondeterministic application to propagate focus alternatives out of islands predicts that association with multiple foci across islands is always unselective. From the perspective of an operator outside the island, two foci inside the island cannot be pulled apart: all the operator can see is one big undifferentiated mass of nondeterminism. Wold 1996, however, emphasizes that association with focus across islands is potentially selective (see also Krifka 1991, 2006, Rooth 1996). For example, the second sentence of (1.21) conveys that both John and Bobby were such that we only saw the entries Hoover made about him. Such a reading requires *only* to selectively associate with *HOOVER*$_F$ (to the exclusion of *BOBBY*$_F$) and *also* to selectively associate with *BOBBY*$_F$ (to the exclusion of *HOOVER*$_F$) across a scope island boundary:

(1.21) We only saw the entries ⟨HOOVER$_{\text{F}}$ made about John⟩.
We also only saw the entries ⟨HOOVER$_{\text{F}}$ made about BOBBY$_{\text{F}}$⟩.
(after Rooth 1996, ex. 43) (*selective association with focus across islands*)

Accounting for these troublesome data requires nothing beyond an application of the monadic credo—i.e. taking a side-effects-based perspective on Rooth's alternative semantics. As with indefiniteness, we first identify an underlying monad to be paired with scopal application. What's needed here is a monad where programs have two dimensions: in one dimension lives vanilla values that compose via functional application, and in the other dimension lives a set of alternative values that compose nondeterministically. The relevant construct is something I call the Focus monad (identical to Shan's 2002 Pointed Powerset monad, also proposed for focus). The Focus monad essentially pairs the Identity monad with the Set monad (for any pair $m = \langle a,\ A\rangle$, $m_0 := a$ and $m_1 := A$):

Definition 1.5 (The Focus monad).

$$\begin{aligned} \mathsf{M}\alpha &::= \alpha \times (\alpha \to t) \\ a^{\eta} &:= \langle a,\ \{a\}\rangle \\ m \multimap k &:= \langle (k\, m_0)_0,\ \bigcup_{a \in m_1} (k\, a)_1 \rangle \end{aligned}$$

It's straightforward to refactor Rooth's proposal for F-marking in these terms. The meaning of an F-mark is a function $\mathbf{F}$ from a value a to a pair of a and the (contextually relevant) alternatives to a:

Definition 1.6 (Semantics of F-marking).

$$a^{\mathbf{F}} := \langle a,\ \mathsf{Alt}\, a\rangle$$

That is all there is to it. Reformulating Rooth's alternative semantics in terms of an underlying monad predicts that the side effects incurred by focus will be able to take bona fide scope out of scope islands: like State.Set side effects, Focus side effects can survive evaluation to take scope beyond the boundaries of a scope island:

Fact 1.6 (Resetting *BILL$_F$ came*).

$$\left(\frac{\mathsf{b}^{\mathsf{F}}_{x} \multimap [\,]}{\mathsf{came}\ x}\right)^{\downarrow\uparrow} = \underbrace{(\mathsf{b}^{\mathsf{F}}_{x} \multimap (\mathsf{came}\ x)^{\eta})^{\uparrow}}_{\text{Focus-impure}} = \frac{\mathsf{b}^{\mathsf{F}}_{x} \multimap [\,]}{\mathsf{came}\ x}$$

Using an underlying Focus monad likewise predicts that selective association with focus across islands is possible. The analysis is totally analogous to the account of selective exceptional scope out of islands in examples such as (1.13). Since the grammar generates higher-order evaluated structures (type $\mathsf{MM}t$), we are able to avoid conflating different sources of Focus side effects outside the boundary of the scope island. See Fact 1.7, the Focus monad analog of Fact 1.3 (the ellipses elide an irrelevant complication due to the semantics of the relative clause gap).

Fact 1.7 (Higher-order evaluated meaning for *HOOVER$_F$ made about BOBBY$_F$*).

$$\mathsf{b}^{\mathsf{F}}_{y} \multimap (\mathsf{h}^{\mathsf{F}}_{x} \multimap (\cdots)^{\eta})^{\eta} :: \mathsf{MM}t$$

In sum, treating focus as a side effect predicts that association with focus is both insensitive to islands and potentially selective across islands. In addition, the account establishes a (so far as I know) novel connection between the ways that indefinites/disjunctions take selective scope out of islands and the way that focus alternatives do so. Finally, I show that no conflicts arise when we use an underlying Focus monad alongside an underlying State.Set monad. The two coexist seamlessly in a single grammar (as they must, given that the Focus monad doesn't obviate the need for the State.Set monad, and vice versa), without any need to complicate the basic compositional machinery.

1.4 The long view

Given the relationship I proposed between underlying monads and scopal composition, the monadic credo turns out to be a powerful organizing principle when constructing a grammar. For indefiniteness, I build on dynamic semantics to offer a theory in terms of state-changing and nondeterministic side effects. For focus, I build on Roothian alternative semantics to offer a theory of focus-induced side effects in terms of the Focus monad. *Whatever* side effects a theorist sees evidence for, the inevitable prediction is that they'll be the sorts of things that survive evaluation and take scope out of scope islands.

To underline this point, the dissertation concludes with a bit of alternate history. I consider what happens when we swap out the underlying State.Set monad (*state-changing* plus nondeterministic side effects) for an underlying Reader.Set monad (*state-sensitive* plus nondeterministic side effects):

Definition 1.7 (The Reader.Set vs. State.Set monads).

$$
\begin{aligned}
\mathsf{M}\alpha &::= s \to \alpha \to t & \qquad \mathsf{M}\alpha &::= s \to (\alpha \times s) \to t \\
a^{\eta} &:= \lambda s.\, \{a\} & a^{\eta} &:= \lambda s.\, \{\langle a,\, s\rangle\} \\
m \multimap k &:= \lambda s. \bigcup_{a \in m\,s} k\; a\; s & m \multimap k &:= \lambda s. \bigcup_{\langle a, s'\rangle \in m\,s} k\; a\; s'
\end{aligned}
$$

Definition 1.8 (Reader.Set "lexicon" sampling).

Item	**Meaning**	**Type**
John	j	e
saw	saw	$e \to e \to t$
and	$(\wedge)$	$t \to t \to t$
a linguist	$\textbf{a.ling} := \lambda s.\, \{x \mid \mathsf{ling}\, x\}$	$\mathsf{M}e$
not	$\textbf{not} := \lambda m s.\, \{\neg\exists a.\, a \in m\, s \wedge a\}$	$\mathsf{M}t \to \mathsf{M}t$
if	$\textbf{if} := \lambda m n.\, \textbf{not}\, (m_p \multimap (\textbf{not}\, n)_q \multimap (p \wedge q)^{\eta})$	$(\mathsf{M}t \to \mathsf{M}t) \to \mathsf{M}t$
every linguist	$\textbf{ev.ling} := \lambda k.\, \textbf{not}\, ((\textbf{a.ling})_x \multimap \textbf{not}\, (k\, x))$	$(e \to \mathsf{M}t) \to \mathsf{M}t$

The Reader.Set monad, just like the State.Set monad, has a nondeterministic core. This means that, so far as exceptionally scoping nondeterminism goes, the Reader.Set monad matches the coverage of the State.Set monad beat for beat. As I noted previously, the Reader.Set monad corresponds to Kratzer & Shimoyama's 2002 state-sensitive, nondeterministic semantics. However, the Reader.Set semantics sidesteps important issues Shan 2004 identifies for Kratzer & Shimoyama's compositional formulation (specifically, Kratzer & Shimoyama's proposed abstraction operation leads to problematic predictions when nondeterministic constituents are bound into). Another point of divergence is that the Reader.Set semantics (like the State.Set semantics) allows for selective exceptional scope-taking out of islands, while Kratzer & Shimoyama's semantics does not.

Where the Reader.Set and State.Set monads crucially differ is their treatment of anaphora. While the Reader monad is all about state-sensitivity (i.e. input), the State monad is all about changing state (i.e. input *and output*). The upshot: moving to the Reader.Set monad means giving up the State.Set monad's account of dynamic anaphora. While the Reader.Set semantics readily handles cases of *in-scope* binding, the Reader monad just isn't built for outputting modified inputs, and so evaluating a scope island means discarding any binding information generated inside the island. The monadic perspective thus illuminates a fundamental and unremarked connection between dynamic and alternative semantics for indefinites (both are nondeterministic, reflected in their reliance on the Set monad), while providing a clear reckoning of the way in which they differ (their divergent perspectives on anaphora correspond to a choice between the Reader and State monads).

Wrapping up. In sum, the analysis to come represents an advance in the theory of exceptional scope. The theory I propose—that exceptional scope results when side effects take scope after evaluation—explains the data reported above (and more besides) in a unified way, with better empirical coverage than its static or dynamic predecessors (even with respect to their more narrowly defined empirical domains), and by means of independently motivated assumptions, while incurring (I argue) no significant stipulations relative to standard dynamic theories of interpretation. In addition to these empirical gains, the results to come have independent theoretical interest. I offer a novel, directly compositional (cf. Jacobson 1999) theory of dynamic semantics *as computation*, one which reveals a thoroughgoing connection between dynamic- and alternatives- based theories of indefinites, and develop new techniques for integrating theories of side effects such as state and nondeterminism with the theory of scope.

Part I

Theory: side effects and scope

Chapter 2

Dynamic side effects

2.1 Introduction

This chapter lays the formal foundations for the rest of the thesis. It has two main goals. First, I introduce *monads*, a technique from category theory and theoretical computer science used to model computational side effects (see e.g. Moggi 1989, Wadler 1994, 1995 for non-linguistic discussion of monads, and Shan 2002, Giorgolo & Unger 2009, Unger 2012, Giorgolo & Asudeh 2012 for linguistic applications of monads). Second, I use monads to build a novel dynamic semantics for anaphora based on state-changing and nondeterministic side effects. As I noted in Chapter 1, my central claim in this dissertation is that exceptional scope happens when side effects take scope after evaluation. Having a side-effects-based theory of dynamic semantics is thus essential for explaining how state-changing and nondeterministic side effects scope out of islands.

Semantic composition in the Fregean tradition is fundamentally about combining meanings by functional application. In a *pure* grammar, that's all that happens: we have functions, we have arguments, we apply functions to arguments, and that is that. But some aspects of linguistic meaning don't fit neatly into that picture. Semanticists sometimes care about state-sensitive meanings—meanings that may return different values depending on when or where they're evaluated. Sometimes we care about nondeterminism—about keeping track of a number of computations at once. Sometimes we care about both at the same time. In each of these cases, something happens in addition to functional application—i.e. keeping track of a state, doing a number of computations in parallel, or a mixture of the two. The things that happens in a semantic computation above and beyond the meat-and-potatoes business of functional application are that computation's *side effects*. Following Shan 2002, I'll introduce a theory of linguistic side effects in terms of monads and apply this perspective to several cases of side effects regimes familiar to linguists.

We'll set aside monads while I introduce a minimal, novel dynamic interpretation scheme that I call DyS. Its main utility is that it distills the principal strain of dynamic-semantic theorizing down to its essence, after which point we can begin to reason about dynamic semantics in terms of side effects. This distillation has two important parts: DyS (i) allows sentences to take inputs, modify them, and then *output* them, (ii) and adopts a *nondeterministic* perspective on meaning in order to treat indefinites as nondeterministic analogs of proper names.

We go in search of a monad for dynamic semantics—i.e. a monad which captures the ability of linguistic expressions to modify the state of a discourse, as well as the ability of linguistic expressions to introduce nondeterminism into a semantic computation. We converge on a monad for modifying and outputting inputs and combine it with a monad for nondeterminism. This combination yields a new type of dynamic semantics, one which treats dynamic effects as proper side effects, connects dynamic semantic theorizing to extant nondeterministic theories of indefinites, and forms the basis for the empirical analyses in subsequent chapters.

2.2 Monads

2.2.1 Roadmap

This section introduces *monads* and applies them to the semantic analysis of natural language. We'll begin by reviewing three common ways semanticists enrich grammars to go beyond functional application. We'll look at state-sensitive functional application, nondeterministic functional application, and a mixture, state-sensitive nondeterministic functional application. Then we'll move to monads as a way to theorize about these extensions in terms of side effects. I'll first introduce monads in the abstract and show how to construct a grammar based on an arbitrary monad. Then I'll get concrete. We'll see monads for state-sensitivity, nondeterminism, and a mix, and we'll see how these monads underly the three enriched grammars.

2.2.2 Functional application with side effects

A proposal with its roots in Frege 1892 takes the meaning of a binary-branching node to be the result of doing functional application ('A') on the denotations of its daughters. See Definition 2.1. To illustrate, if *John* denotes an individual j (type e) and *left* a function from individuals to booleans left (type $e \to t$), $[\![\text{John left}]\!]$ is given by $\mathsf{A}\,[\![\text{John}]\!]\,[\![\text{left}]\!] = \mathsf{left}\,\mathsf{j}$ (type t).[1]

Definition 2.1 (Functional application).

$$\mathsf{A}\,x\,y := x\,y \text{ or } y\,x, \text{ whichever is defined}$$

This way of stating A *overloads* it: whether A performs forward or backward functional application depends on which of $\{x, y\}$ is typed as the functor. This *type-driven* approach to interpretation (e.g. Klein & Sag 1985) streamlines definitions considerably. We'll see it a lot in what follows.

Modes of composition that properly extend A are common. We'll run through three examples.

1 Some notational conventions. Application is left-associative: e.g. $a\,(b\,c)\,d$ abbreviates $((a(b(c)))(d))$. Types are *right*-associative: e.g. $\sigma \to (\tau \to \zeta) \to \chi$ abbreviates $\sigma \to ((\tau \to \zeta) \to \chi)$ and is equivalent to $\langle\sigma, \langle\langle\tau, \zeta\rangle, \chi\rangle\rangle$. Additionally, $\lambda xy.\, f$ abbreviates $\lambda x.\, \lambda y.\, f$; the period extends the scope of the preceding λ as far to the right as possible, i.e. while still yielding a well-formed λ-term (these conventions also apply to $\forall$ and $\exists$). Text sans serifs names non-monadic model-theoretic values, while **boldface sans** is used to name monadic model-theoretic objects and functions over monadic objects. Parens are systematically omitted when doing so doesn't create ambiguity.

First, meanings are often treated as functions from sequences of *states*—say, worlds, times, utterance contexts, assignment functions, or combinations thereof—into values (e.g. Montague 1974, Heim & Kratzer 1998). When meanings are parametrized to a state, the interpretation of a binary-branching node is gotten by doing *state-sensitive functional application* ('SSA') on the denotations of its daughters, see Definition 2.2. SSA reads in a state, uses it to determine a couple values, and finally combines the resulting values via functional application. As an example, if *she* denotes some state-sensitive m (type $s \to e$) and *left* a state-sensitive $\lambda s.\,\mathsf{left}$ (type $s \to e \to t$), $[\![\text{she left}]\!]$ is given by $\mathsf{SSA}\,[\![\text{she}]\!]\,[\![\text{left}]\!] = \lambda s.\,\mathsf{left}\,(m\,s)$ (type $s \to t$).

Definition 2.2 (State-sensitive functional application).

$$\begin{aligned}\mathsf{SSA}\,m\,n :=\ & \lambda s.\,\mathsf{A}\,x\,y\\ & \text{for } x := m\,s\\ & \text{for } y := n\,s\end{aligned}$$

The *for*-notation used here is a convention that makes some expressions (especially later in this chapter) easier to parse. *For*-statements appear below an expression with one or more unbound variables and tells us how those variables are to be understood. For example, $\mathsf{SSA}\,m\,n$ is equivalent to $\lambda s.\,m\,s\,(n\,s)$.

A second way to extend A is seen in grammars where meanings are *sets* of values (e.g. Hamblin 1973). This models *nondeterminism* or *randomness* by allowing the interpretation function to traverse a number of computations in parallel. The interpretation of a binary-branching node on this picture is given by performing *nondeterministic functional application* ('NA') on the denotations of its daughters. NA takes a set of functions and a set of arguments and gives back a set whose members are the results of combining every member of the former with every member of the latter, again via A. See Definition 2.3. As an example, if *a donkey* denotes a set of individuals $\{x \mid \mathsf{donkey}\,x\}$ (type $e \to t$) and *left* a singleton set of properties $\{\mathsf{left}\}$ (type $(e \to t) \to t$), $[\![\text{a donkey left}]\!]$ is given by $\mathsf{NA}\,[\![\text{a donkey}]\!]\,[\![\text{left}]\!] = \{\mathsf{left}\,x \mid \mathsf{donkey}\,x\}$ (type $t \to t$).[2]

Definition 2.3 (Nondeterministic functional application).

$$\begin{aligned}\mathsf{NA}\,m\,n := \{\,\mathsf{A}\,x\,y \mid\ & x \in m\ \wedge\\ & y \in n\,\}\end{aligned}$$

As a final example, we can combine SSA and NA into an operation which reads in a parameter, distributes it across two state-sensitive meanings, and does NA on the resulting sets, as in Definition 2.4. Call this operation *state-sensitive nondeterministic functional application* ('SSNA'). SSNA is the interpretation rule given in Kratzer & Shimoyama 2002, Alonso-Ovalle 2006.[3] Suppose

2 The *set-builder notation* works as follows: $\{\Phi \mid \Psi\}$ is the set containing all and only the Φ's meeting some condition Ψ. E.g. $\{x \mid \mathsf{donkey}\,x\}$ is the set S such that $x \in S$ iff x is a donkey—i.e. it names the set of donkeys.

3 A reverse layering, i.e. with sets of state-sensitive meanings and the implied operation of nondeterministic state-sensitive functional application (NSSA), will be discussed in Chapter 5.

for illustration that *a man* means $\lambda s.\,\{x \mid \mathsf{man}\,x\}$ and *met her mom* means $\lambda s.\,\{\mathsf{met}\,(\mathsf{mom}\,(m\,s))\}$. Then ⟦a man met her mom⟧ is given by SSNA ⟦a man⟧ ⟦met her mom⟧ $= \lambda s.\,\{\mathsf{met}\,(\mathsf{mom}\,(m\,s))\,x \mid \mathsf{man}\,x\}$.

Definition 2.4 (State-sensitive nondeterministic functional application).

$$\begin{aligned}\mathsf{SSNA}\,m\,n \coloneqq \lambda s.\,\{\,\mathsf{A}\,x\,y \mid {} & x \in m\,s \wedge \\ & y \in n\,s\,\}\end{aligned}$$

Each of SSA, NA, and SSNA represents a proper extension of A: A still happens *eventually*, but in each case something else has to happen first. Cribbing some computer science terminology, we'll say that things in the domains of SSA, NA, and SSNA incur *side effects*—i.e. those something elses that have to be dealt with before A applies.

Semanticists are generally content to reckon with data that seem to require extensions of A (and thus side effects) by stipulating an enriched semantics for composition and rewriting the grammar wholesale so that some lexical entries (usually not all or even most) can take advantage of the extra power that the extended mode of composition affords. In what follows, I will (following Shan 2002) explore an alternative way to construct grammars, one which gives a general recipe for composition in the presence of arbitrary side effects. Put another way, the theory of composition I develop is modular; it has a single free parameter which can be fixed in different ways to give A, SSA, NA, or SSNA as the semantics of binary combination. That free parameter is a *monad*. This is where we turn now.

2.2.3 What's in a monad

This section introduces monads (e.g. Moggi 1989, Filinski 1994, Wadler 1994, 1995) as a way to systematically turn pure grammars (i.e. grammars limited to A) into grammars capable of handling programs that incur side effects (e.g. grammars where composition is via SSA, NA, or SSNA). Monads are a tool developed by computer scientists for reasoning about side effects in pure programming languages such as Haskell or the simply-typed lambda calculus. Following Shan 2002, I formulate a *monadic grammar* that treats semantic composition as program composition. The basic structure of the monadic grammar is invariant; different ways of fixing its "underlying monad" will yield different compositional regimes. Signpost: this section is of necessity somewhat abstract and formally involved. On a first pass, the reader may wish to skim, get their bearings with concrete examples in Section 2.3, and then return.

Formally, a monad is a triple $\langle \mathsf{M}, \eta, \multimap \rangle$ of a *type constructor* M, an *injection function* η, and a *sequencing operation* $(\multimap)$. $\mathsf{M}\alpha$ specifies the type of *programs* which return α-typed values.[4] For example, $\mathsf{M}e$ is the type of a program that returns an individual, and $\mathsf{M}t$ the type of a program that returns a boolean. The other members of the triple give the monad its semantic teeth. The injection

4 M, η, and $(\multimap)$ should always be understood as parametrized, either to an arbitrary monad or to whatever monad I'm discussing at the moment (it'll always be obvious from the context how to understand them).

operator η gives a recipe for upgrading a value into a trivial/pure program (that is, one without any side effects). Thus its type is $\alpha \to \mathsf{M}\alpha$ (for some value type α). For example, j^η is a trivial program of type $\mathsf{M}e$ that returns j, $(\mathsf{left}\,\mathsf{j})^\eta$ a trivial program of type $\mathsf{M}t$ that returns True iff John left, and so on. Finally, $m \multimap \lambda v.\, \pi$ (pronounced, "run m to determine v in π") specifies how to evaluate a program m and bind the resulting value to v in π. Fixing two types α and β, $(\multimap)$ has type $\mathsf{M}\alpha \to (\alpha \to \mathsf{M}\beta) \to \mathsf{M}\beta$.

We can save symbols and get more intuitive formulas if we adopt a slightly abbreviated notation for sequencing, as in Definition 2.5. The notation just replaces a lambda abstract on the right side of a sequencing operator with a subscript on the program on the left. This convention is inspired by the programming language Haskell's do-notation (Haskell has a native syntax for monadic computation), but it also—not coincidentally—bears a strong resemblance to subscripting conventions used in linguistics (see e.g. Heim 1998, and especially Büring's 2005 operation of Index Transfer).

Definition 2.5 (Abbreviating sequencing).

$$m_v \multimap \pi := m \multimap \lambda v.\, \pi$$

Any putatively monadic $\langle \mathsf{M}, \eta, \multimap \rangle$ has has to obey three *monad laws* to pass muster. These are given in Definition 2.6. LeftID requires that building a trivial program out of a value a and sequencing that program with $\lambda v.\, \pi$ is the same as foregoing the injection and just binding v to a in π.[5] RightID ensures that sequencing any program m with the injection function η gives back m. LeftID and RightID together ensure that η and $\multimap$ neither wantonly introduce nor carelessly discard information (i.e. values or side effects). Finally, Assoc requires that sequencing m with $\lambda v.\, \pi$ and sequencing the resulting program with $\lambda u.\, \theta$ is equivalent to sequencing m with a single function which can be determined from $\lambda v.\, \pi$ and $\lambda u.\, \theta$ (this latter function, $\lambda v.\, \pi_u \multimap \theta$, is the result of function-composing $\lambda n.\, n_u \multimap \theta$ with $\lambda v.\, \pi$).[6] Assoc ensures that the sequencing operation works like a one-dimensional *pipeline* for side effects built on top of the operations on values: whether a pipeline's sections are built sequentially or concurrently, the upshot should be the same when you turn on the faucet. In other words, *order of evaluation*, not hierarchy, is what matters for side effects.

Definition 2.6 (The monad laws).

$$\begin{aligned} a^\eta_v \multimap \pi &= \pi[a/v] && \text{LeftID} \\ m_v \multimap v^\eta &= m && \text{RightID} \\ (m_v \multimap \pi)_u \multimap \theta &= m_v \multimap \pi_u \multimap \theta && \text{Assoc} \end{aligned}$$

2.2.4 Monadic application and linear interpretation

Any monad admitting programs of type $\mathsf{M}\alpha$ and $\mathsf{M}(\alpha \to \beta)$ (for any α and β) determines a notion of *monadic functional application*, given in Definition 2.7. $\mathbf{A}$ combines a program of type $\mathsf{M}\alpha$ with a

5 I write '$\pi[a/v]$' for the formula just like π but with a substituted for all free occurrences of v.
6 The sequencing notation is right-associative: $m_v \multimap \pi_u \multimap \theta = m_v \multimap (\pi_u \multimap \theta)$.

program of type $\mathsf{M}(\alpha \to \beta)$ and returns a program with type $\mathsf{M}\beta$. $\mathbf{A}\,m\,n$ means evaluating m and n in sequence and doing functional application on the resulting values, threading the side effects from m to n to the composite program (notice that since A is overloaded, so is $\mathbf{A}$):[7]

Definition 2.7 (Monadic application).

$$\mathbf{A}\,m\,n \coloneqq m_x \multimap n_y \multimap (\mathsf{A}\,x\,y)^\eta$$

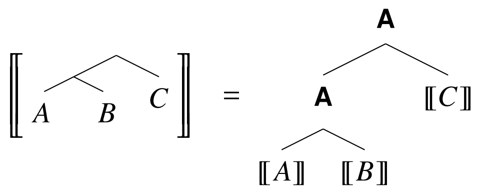
Familiarly, then, the monadic interpretation of binary-branching nodes involves monadically applying the meaning of one branch to the meaning of the other. See Definition 2.8, which we can gloss as "run $[\![L]\!]$, then $[\![R]\!]$, and return a program that does functional application on the resulting values":

Definition 2.8 (Monadically interpreting binary-branching nodes).

$$[\![L\,R]\!] \coloneqq \mathbf{A}\,[\![L]\!]\,[\![R]\!]$$

Two facts are worth mentioning at this early point: (i) For any binary-branching tree $\mathcal{T}$, $[\![\mathcal{T}]\!]$ yields a semantic "analysis tree" with the same structure as $\mathcal{T}$. See (2.1) for an example (where I've de-Curry'd $\mathbf{A}$ to emphasize the parallel). Because of this close correspondence between syntactic structures and analysis trees, I'll generally only give a detailed rendering of the latter; the former can be readily inferred from it (see Appendix A for more details about the sorts of constituent structures I assume).

(2.1)



(ii) Together, Definitions 2.7 and 2.8 entail *left-to-right evaluation*: for any binary-branching node, the constituent on the left is evaluated before the constituent on the right. In other words, the right constituent is evaluated in a context that includes any side effects incurred by the constituent on the left. Whether this linear bias matters will depend on the monad underlying $\mathbf{A}$. We will see examples of both commutative and non-commutative monads in what follows.

2.2.5 Rebracketing and the side effects pipeline

An important consequence of the monad laws, in particular Assoc, is that hierarchical notions (e.g. c-command) are *irrelevant for side effect propagation*. If X is to the left of Y, the side effects incurred by X will (given that our rule for interpretation stipulates left-to-right evaluation) *always* inform the evaluation of Y, *regardless of how embedded X might be* relative to Y.

7 Combining side effects with predicative interpretive schemes such as neo-Davidsonian event semantics (e.g. Parsons 1990) is straightforward. If a monad $\mathcal{M}$ admits programs of type $\mathsf{M}(\alpha \to t)$, $\mathcal{M}$ determines a notion of *monadic predicate modification*, with type $\mathsf{M}(\varsigma \to t) \to \mathsf{M}(\varsigma \to t) \to \mathsf{M}(\varsigma \to t)$ (ς might be the type of eventualities): $m_v \multimap n_u \multimap (v \cap u)^\eta$.

To see this, it's helpful to begin with Fact 2.1, a theorem I call Rebracket. Rebracket says that sequencing the interpretation of a binary-branching node with any $\lambda v.\, \pi$ is equivalent to evaluating the denotation of the left node m to give a value x, then evaluating the denotation of the right node n to give a value y, doing functional application on x and y, and binding v to $\mathsf{A}\, x\, y$ in π.

Fact 2.1 (Rebracket).

$$(\mathbf{A}\, m\, n)_v \multimap \pi = m_x \multimap n_y \multimap \pi[\mathsf{A}\, x\, y/v]$$

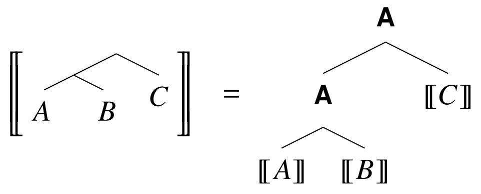
Proof.

$$\begin{aligned}
(\mathbf{A}\, m\, n)_v \multimap \pi &= (m_x \multimap n_y \multimap (\mathsf{A}\, x\, y)^\eta)_v \multimap \pi && \mathbf{A}\\
&= m_x \multimap (n_y \multimap (\mathsf{A}\, x\, y)^\eta)_v \multimap \pi && \text{Assoc}\\
&= m_x \multimap n_y \multimap ((\mathsf{A}\, x\, y)^\eta)_v \multimap \pi && \text{Assoc}\\
&= m_x \multimap n_y \multimap \pi[\mathsf{A}\, x\, y/v] && \text{LeftID}
\end{aligned}$$

□

Rebracket and the left-biased $\llbracket\cdot\rrbracket$ entail that any analysis tree rooted in a syntactic tree $\mathcal{T}$ can be equivalently reformulated in terms of (i) a sequence of programs recapitulating the linear order of the corresponding terminals in $\mathcal{T}$ and (ii) a trivial program that performs functional application on values. An illustration for (2.1) is given in (2.2). A completely left-branching phrase marker generates an isomorphic analysis tree, but a single application of Rebracket shows us that the result is equivalent to a fully linearized analysis, where we run the programs denoted by the terminal nodes in their linear order and do functional application twice on the resulting values. In particular, though A doesn't c-command C in $\mathcal{T}$, $\llbracket A\rrbracket$ is evaluated before $\llbracket C\rrbracket$ in $\llbracket \mathcal{T}\rrbracket$; simply by virtue of being to C's left, A's side effects are incurred first.

(2.2)



$$\begin{aligned}
&= (\mathbf{A}\, \llbracket A\rrbracket\, \llbracket B\rrbracket)_v \multimap \llbracket C\rrbracket_c \multimap (\mathsf{A}\, v\, c)^\eta && \mathbf{A}\\
&= \llbracket A\rrbracket_a \multimap \llbracket B\rrbracket_b \multimap \llbracket C\rrbracket_c \multimap (\mathsf{A}\, (\mathsf{A}\, a\, b)\, c)^\eta && \text{Rebracket}
\end{aligned}$$

A similar pattern holds for any well-typed $\llbracket \mathcal{T}\rrbracket$. Regardless of the syntactic bracketing in $\mathcal{T}$ (and regardless of its function-argument structure), the result will, given Rebracket, be equivalent to evaluating the terminals in $\mathcal{T}$ in their linear order and then doing functional application on the resulting values. It's useful in this regard to consider the interpretation assigned to a completely right-branching $\mathcal{T}$, namely $\llbracket A\ [B\ C]\rrbracket = \llbracket A\rrbracket_a \multimap \llbracket B\rrbracket_b \multimap \llbracket C\rrbracket_c \multimap (\mathsf{A}\, a\, (\mathsf{A}\, b\, c))^\eta$. The side effect pipeline remains unchanged; the only difference is the function-argument structure. Since Rebracket follows from $\mathbf{A}$, Assoc, and LeftID, *any* monadic grammar whose $\llbracket\cdot\rrbracket$ conforms to Definition 2.8 will inherit this remarkable property.

2.2.6 Coercion and constructing derivations

It's natural to suppose that there are different lexical grades of monadic involvement. Some things necessarily depend on the state (e.g. pronouns in SSA-style grammars); others don't (e.g. names). Some things are necessarily nondeterministic (e.g. indefinites in NA-style grammars); others aren't (e.g. lexical verbs). The usual way semanticists deal with this asymmetry is by eliminating it *in the lexicon*, i.e. treating everything as lexically state-sensitive, lexically set-denoting, etc. On the monadic perspective, there is no need to lexically generalize to the worst case. Some things will lexically denote programs; others will lexically denote values. Our policy throughout will be for lexical entries to have the lowest type compatible with their side effects (or lack thereof) and to *coerce* values into trivial programs via an injection type-shifter whose application is recorded via an object-language superscript:

Definition 2.9 (Monadic coercion).

$$[\![X^\eta]\!] \coloneqq [\![X]\!]^\eta$$

For composition to proceed smoothly, the daughters of every binary-branching node must have types $\mathsf{M}\alpha$ and $\mathsf{M}(\alpha \to \beta)$, for some α and β. This means any values will need to be coerced into trivial programs via applications of η. Once this is done, we'll construct an analysis tree via the syntactic parse and $[\![\cdot]\!]$, Rebracket into a linear sequence of programs evaluated in the order of the corresponding terminals, sweep the trivial programs away via LeftID, and, finally, iteratively simplify via the meanings of η and $(\multimap)$.

2.3 Examples

We'll use a few different monads to add side effects to a bare-bones applicative grammar. After warming up with a monad which encapsulates trivial side effects, we'll move on to monads for state-sensitivity, nondeterminism, and a combination of the two. To wit, A corresponds to the Identity monad, SSA to the Reader monad, NA to the Set monad, and SSNA to the Reader.Set monad. This demonstrates that a side-effects-based perspective on semantic composition *already* underlies much semantic theorizing, whether we attend to it or not.

2.3.1 Pure programs: the Identity monad

Our first example is the Identity monad. Programs in the Identity monad have *no* side effects (alternatively, they have only trivial side effects). The Identity monad's type constructor is the identity function on types, its η operation is the identity function on values (i.e. programs in the Identity monad are indistinguishable from values), and sequencing m with $\lambda v.\,\pi$ is just binding v to m in π (i.e. β-reduction/functional application):[8]

8 Proofs that the monads considered in the text satisfy the monad laws are given in Appendix B.

Definition 2.10 (The Identity monad).

$$\begin{aligned}\mathsf{M}\alpha &::= \alpha \\ a^{\eta} &:= a \\ m_v \multimap \pi &:= \pi[m/v]\end{aligned}$$

Since the Identity monad lacks side effects, monadic application—running two programs in sequence and doing A on the resulting values—is just A all over again. So the grammar determined by the Identity monad represents a *trivial* extension of the basic applicative grammar.

Fact 2.2 (Application in the Identity monad).

$$\begin{aligned}\mathbf{A}\,m\,n &= \mathsf{A}\,x\,y \\ &\quad \text{for } x := m \\ &\quad \text{for } y := n\end{aligned}$$

Proof.

$$\begin{aligned}\mathbf{A}\,m\,n &= m_v \multimap n_u \multimap (\mathsf{A}\,v\,u)^{\eta} && \mathbf{A} \\ &= m_v \multimap n_u \multimap \mathsf{A}\,v\,u && \eta \\ &= m_v \multimap \mathsf{A}\,v\,n && \multimap \\ &= \mathsf{A}\,m\,n && \multimap\end{aligned}$$

□

Though the Identity monad seems to hardly bear mentioning, we will see in Chapter 3 that it plays an important, though implicit, role in extant *continuations*-based analyses of scopal phenomena in natural language.

2.3.2 State-sensitive programs: the Reader monad

The monad underlying SSA is called the Reader monad. Programs in the Reader monad are functions from a state to a value. Accordingly, the type of a program yielding a value of type α is $s \to \alpha$ (remaining agnostic for now about what sorts of states inhabit type s). So a monadic individual (type $\mathsf{M}e$) corresponds to an individual concept, and a monadic boolean (type $\mathsf{M}t$) to a proposition (i.e. the characteristic function of some set of states). Lifting a value into the Reader monad means taking a value a and returning a constant function from stacks to a (i.e. a rigid designator; η is an instance of the K combinator from combinatory logic). Finally, sequencing m with $\lambda v.\,\pi$ means (i) using a state s to retrieve a value (i.e. $m\,s$), (ii) binding v to $m\,s$ in π, and (iii) sharing s with the result.[9]

9 It's no coincidence that $(\multimap)$, i.e. $\lambda mks.\,k\,(m\,s)\,s$, turns out to be (a reordered version of) Jacobson's 1999 **z**-shifter. Jacobson's variable-free semantics can be thought of as implementing something like a Reader-monadic perspective on meaning, where dependencies on a state are replaced by dependencies on individuals.

Definition 2.11 (The Reader monad).

$$\begin{aligned} \mathsf{M}\alpha &::= s \to \alpha \\ a^\eta &:= \lambda s.\, a \\ m_v \multimap \pi &:= \lambda s.\, \pi[ms/v]\, s \end{aligned}$$

Monadic application is evaluating two programs in sequence and doing functional application on the resulting values. In the Reader monad, this means distributing a state to two state-sensitive programs and doing functional application on the values that result, as in Fact 2.3. This operation is identical to SSA, justifying our claim that the Reader monad underlies the move from an applicative to a state-sensitive grammar (as also claimed by Shan 2002).

Fact 2.3 (Application in the Reader monad).

$$\begin{aligned} \mathbf{A}\, m\, n = {} & \lambda s.\, \mathsf{A}\, x\, y \\ & \text{for } x := m\, s \\ & \text{for } y := n\, s \end{aligned}$$

Proof.

$$\begin{aligned} \mathbf{A}\, m\, n &= m_v \multimap n_u \multimap (\mathsf{A}\, v\, u)^\eta && \mathbf{A} \\ &= m_v \multimap n_u \multimap \lambda s.\, \mathsf{A}\, v\, u && \eta \\ &= m_v \multimap \lambda s.\, \mathsf{A}\, v\, (n\, s) && \multimap \\ &= \lambda s.\, \mathsf{A}\, (m\, s)\, (n\, s) && \multimap \end{aligned}$$

□

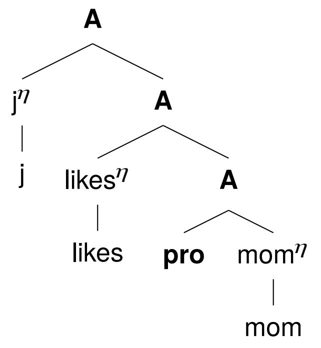
A Reader-monadic meaning for pronouns is given in Definition 2.12. (We'll abstract away from the precise semantics of $s_\top$ for the time being. For now just take it to be a black box for selecting a referent from s.) This meaning is irreducibly monadic (thus, impure); it cannot be derived from a value via an application of η. Sequencing **pro** with $\lambda v.\, \pi$ means evaluating **pro** at a state s and binding the resulting value to v in π: i.e. $\textbf{pro}_v \multimap \pi = \lambda s.\, \pi[s_\top/v]\, s$.

Definition 2.12 (Pronouns in the Reader monad).

$$\begin{aligned} [\![\text{s/he}]\!] &:= \textbf{pro} \\ &:= \lambda s.\, s_\top \end{aligned}$$

Here is an analysis of *John likes her mom* deriving an unbound reading of the pronoun, i.e. where its reference is fixed by the state at which the sentence is evaluated. See (2.3). A few words on notation and derivation construction: Here and throughout, singly-branching nodes record the application of a unary rule (e.g. the η shift), and binary branches record the application of a binary rule (e.g. monadic application). To simplify things, I'll always treat the genitive morpheme as semantically vacuous and model the meanings of relational nouns like *mom* as type $e \to e$ functions (e.g. from individuals to their mothers). The way this derivation goes is typical. After lifting the values into the monad and doing monadic application, we Rebracket, simplify the trivial programs away with LeftID, and apply η and $(\multimap)$. The result is as expected, a non-constant function from states to booleans (assuming John likes some moms and not others).

(2.3) $[\![\text{John}^\eta \text{ likes}^\eta \text{ her mom}^\eta]\!] =$



$$
\begin{aligned}
&= \mathsf{j}^\eta_x \multimap \mathsf{likes}^\eta_f \multimap \textbf{pro}_y \multimap \mathsf{mom}^\eta_g \multimap (f\,(g\,y)\,x)^\eta && \text{Rebracket}\\
&= \textbf{pro}_y \multimap (\mathsf{likes}\,(\mathsf{mom}\,y)\,\mathsf{j})^\eta && \text{LeftID}\\
&= \textbf{pro}_y \multimap \lambda s.\,\mathsf{likes}\,(\mathsf{mom}\,y)\,\mathsf{j} && \eta\\
&= \lambda s.\,\mathsf{likes}\,(\mathsf{mom}\,s_\top)\,\mathsf{j} && \multimap
\end{aligned}
$$

In addition to the unbound reading, we can derive readings with binding by fiddling with the meaning of the VP, as in Definition 2.13 (an analog of Partee's 1973 Derived VP rule, Heim & Kratzer's 1998 predicate abstraction operation, Jacobson's 1999 **z**-shifter, Büring's 2005 β-binding rule, and so on). There's some unfamiliar notation here. For now, assume that $\widehat{sa}$ is a way to "make a available in s", such that $\widehat{sa}_\top = a$. (We'll flesh this out in Section 2.4). Notice that replacing $\widehat{sa}$ with s in Definition 2.13 gives an expression equivalent to $[\![X]\!]$; the only effect of $[\![X]\!]^\triangleleft$ is to modify the state at which $[\![X]\!]$ is evaluated.

Definition 2.13 (Derived VPs in the Reader monad).

$$
\begin{aligned}
[\![X^\triangleleft]\!] &\coloneqq [\![X]\!]^\triangleleft\\
&\coloneqq \lambda sa.\,[\![X]\!]\,\widehat{sa}\,a
\end{aligned}
$$

This means e.g. that $[\![\text{likes}^\eta \text{ her mom}^\eta]\!]^\triangleleft = \lambda sa.\,\mathsf{likes}\,(\mathsf{mom}\,\widehat{sa}_\top)\,a = \lambda sa.\,\mathsf{likes}\,(\mathsf{mom}\,a)\,a$. The result is a state-invariant (i.e. pure) $\mathsf{M}(e \to t)$ that covalues the argument of mom and the external argument of likes; the Derived VP shift effects binding. Sequencing either p^η or a Reader-monadized generalized quantifier Q^η with this yields the familiar bound readings, i.e. $\lambda s.\,\mathsf{likes}\,(\mathsf{mom}\,\mathsf{p})\,\mathsf{p}$ or $\lambda s.\,\mathsf{Q}\,(\lambda a.\,\mathsf{likes}\,(\mathsf{mom}\,a)\,a)$.

The Reader-monadic grammar thus entails that X can only bind Y in $\mathcal{T}$ if X is an argument of some predicate P and Y is or is embedded in some argument of P. In other words, X must c-command Y in $\mathcal{T}$ for binding to be possible. An operator like $\triangleleft$ can modify the state at which a program π *in its scope* is evaluated, but π is the only thing that sees the modified state. This very local sort of local state modification mirrors the way variable binding works in static frameworks for anaphora, where the only possible binding relationships are between semantic arguments of a single predicate (possibly an abstract predicate created via covert movement).

2.3.3 Nondeterministic programs: the Set monad

On to nondeterministic grammars and NA. The monad that models nondeterministic side effects is the Set monad. Nondeterministic programs—i.e. programs that return more than one value—are useful when we'd like to potentially consider a number of computations in parallel. So the type of a Set-monadic program returning a value of type α, i.e. $\mathsf{M}\alpha$, is a *set* of meanings of type α, i.e. $\alpha \to t$. The Set monad's injection operator creates a trivial program out of a value a by mapping it into the singleton set $\{a\}$. Finally, sequencing m with $\lambda v.\, \pi$ means, for each $a \in m$, binding a to v in π, then collecting all of the resulting values into a big set.

Definition 2.14 (The Set monad).

$$\begin{aligned} \mathsf{M}\alpha &::= \alpha \to t \\ a^{\eta} &:= \{a\} \\ m_v \multimap \pi &:= \bigcup_{a \in m} \pi[a/v] \end{aligned}$$

A note on the notation: $\bigcup_{\Phi}\Psi$ is the *infinitary union* of all the sets Ψ meeting some condition Φ. See Definition 2.15. In the case of sequencing, this means $m_v \multimap \pi$ is given by finding each of the sets $\pi[a/v]$ such that $a \in m$ and unioning them into one big set: $x \in m_v \multimap \pi$ iff $\exists a.\, a \in m \wedge x \in \pi[a/v]$.

Definition 2.15 (Infinitary union).

$$\bigcup_{\Phi}\{\xi\} := \bigcup\{\{\xi\} \mid \Phi\} := \{\xi \mid \Phi\}$$

As ever, monadic application means evaluating two programs and doing functional application on the resulting values. Relative to the Set monad, this means doing functional application once for each value in m and each value in n. Monadic application in the Set monad is thus NA.

Fact 2.4 (Application in the Set monad).

$$\begin{aligned} \mathbf{A}\, m\, n = \{\, \mathsf{A}\, x\, y \mid x \in m \wedge \\ y \in n\, \} \end{aligned}$$

Proof.

$$\begin{aligned} \mathbf{A}\, m\, n &= m_v \multimap n_u \multimap (\mathsf{A}\, v\, u)^{\eta} && \mathbf{A} \\ &= m_v \multimap n_u \multimap \{\mathsf{A}\, v\, u\} && \eta \\ &= m_v \multimap \{\mathsf{A}\, v\, y \mid y \in n\} && \multimap \\ &= \{\mathsf{A}\, x\, y \mid x \in m \wedge y \in n\} && \multimap \end{aligned}$$

$\square$

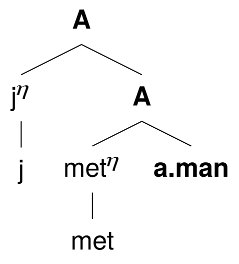
Trivial programs in the Set monad are singleton sets, the result of applying η to some value. Non-trivial programs, naturally enough, are non-singleton sets. By way of illustration, let's suppose for the moment that the meaning of *a man* is the set of men, as in Definition 2.16.

Definition 2.16 (Indefinites in the Set monad).

$$[\![\text{a man}]\!] \coloneqq \textbf{a.man} \coloneqq \{x \mid \mathsf{man}\, x\}$$

As an example, let's look at the Set-monadic analysis of *John met a man*. We lift the value-denoting nodes into trivial programs via an application of η (again, this contrasts with NA grammars, which lexically generalize to the worst case in supposing that all lexical meanings are sets) and combine everything up via applications of **A**. Applying Rebracket a couple times bring us to a linear sequence of programs, which we simplify using LeftID and the Set monad's η and $(\multimap)$.

(2.4) $[\![\text{John}^\eta\ \text{met}^\eta\ \text{a man}]\!] =$



$$\begin{aligned}
&= \mathsf{j}^\eta_x \multimap \mathsf{met}^\eta_f \multimap \textbf{a.man}_y \multimap (f\ y\ x)^\eta && \text{Rebracket}\\
&= \textbf{a.man}_y \multimap (\mathsf{met}\, y\, \mathsf{j})^\eta && \text{LeftID}\\
&= \textbf{a.man}_y \multimap \{\mathsf{met}\, y\, \mathsf{j}\} && \eta\\
&= \{\mathsf{met}\, y\, \mathsf{j} \mid \mathsf{man}\, y\} && \multimap
\end{aligned}$$

The returned set has type $\mathsf{M}t$. Though it can have at most two members, True and False, this is all we need to calculate truth conditions relative to some extensional model: a structure $\mathcal{T}$ will be *true* just in case True is among the values returned by $[\![\mathcal{T}]\!]$.

2.3.4 State-sensitive nondeterministic programs: the Reader.Set monad

We've seen monads underlying A, SSA, and NA. The last example we'll consider before shifting gears to look at dynamic semantics is the monad underlying SSNA—i.e. a monad for programs with both state-sensitive and nondeterministic side effects. This is the Reader.Set monad. The type of a program in the Reader.Set monad is a function from a state to a set of values. E.g. $\mathsf{M}e = s \to e \to t$ and $\mathsf{M}t = s \to t \to t$. The Reader.Set monad's injection operation η takes a value a and returns a constant function from a state to a singleton set containing a, and its sequencing operation feeds a program a state s to extract a set of values, feeds each of these values to the rest of the program (along with s), and collects the resulting sets of values into one big set.

Definition 2.17 (The Reader.Set monad).

$$\begin{aligned} \mathsf{M}\alpha &::= s \to \alpha \to t \\ a^{\eta} &:= \lambda s.\, \{a\} \\ m_v \multimap \pi &:= \lambda s. \bigcup_{a \in m\, s} \pi[a/v]\, s \end{aligned}$$

I didn't pull the Reader.Set monad out of thin air. In fact, the way to wrap Reader functionality around the Set monad is fully determined by what's called a *monad transformer* (Liang et al. 1995). A monad transformer (or monad morphism) is a function from an input monad to an enriched monad—in other words, a recipe for combining two regimes for handling side effects. Relevantly, the Reader monad has a monad transformer, ReaderT, a function from a monad $\mathcal{M}_1 = \langle \mathsf{M}_1, \eta_1, \multimap_1 \rangle$ into a new monad $\mathcal{M}_2 = \langle \mathsf{M}_2, \eta_2, \multimap_2 \rangle$ which wraps Reader-monadic functionality around $\mathcal{M}_1$, as in Definition 2.18. For ReaderT, $\mathcal{M}_2$ is a side effects regime for first doing the Reader thing, and then doing the $\mathcal{M}_1$ thing. It's straightforward to check that applying ReaderT to the Set monad yields the Reader.Set monad.[10] (Notice that applying ReaderT to the Identity monad yields the Reader monad.)

Definition 2.18 (The ReaderT monad transformer).

$$\begin{aligned} \mathsf{M}_2\alpha &:= s \to \mathsf{M}_1\, \alpha \\ a^{\eta_2} &:= \lambda s.\, a^{\eta_1} \\ m_v \multimap_2 \pi &:= \lambda s.\, (m\, s)_u \multimap_1 \pi[u/v]\, s \end{aligned}$$

On to monadic application. Evaluating two Reader.Set programs in sequence and doing functional application on the resulting values means using a state to extract two sets of values and then doing NA on these sets. This amounts to SSNA; the Reader.Set monad underwrites state-sensitive nondeterministic functional application.

Fact 2.5 (Application in the Reader.Set monad).

$$\begin{aligned} \mathbf{A}\, m\, n = \lambda s.\, \{\, \mathsf{A}\, x\, y \mid x \in m\, s \,\wedge \\ y \in n\, s \,\} \end{aligned}$$

Proof.

$$\begin{aligned} \mathbf{A}\, m\, n &= m_v \multimap n_u \multimap (\mathsf{A}\, v\, u)^{\eta} && \mathbf{A} \\ &= m_v \multimap n_u \multimap \lambda s.\, \{\mathsf{A}\, v\, u\} && \eta \\ &= m_v \multimap \lambda s.\, \{\mathsf{A}\, v\, y \mid y \in n\, s\} && \multimap \\ &= \lambda s.\, \{\mathsf{A}\, x\, y \mid x \in m\, s \wedge y \in n\, s\} && \multimap \end{aligned}$$

□

10 Monad transformers also come with a *Lift* operation for turning $\mathsf{M}_1\alpha$ programs into $\mathsf{M}_2\alpha$ programs. Though not directly relevant here, we'll discuss Lift in Chapter 3.

Nontrivial programs in the Reader.Set monad will have state-sensitive side effects, nondeterministic side effects, or both. Indefinite DPs, for instance, will denote *constant* functions from stacks into *non-singleton* sets of values, as in Definition 2.19 (i.e. indefinites have only nondeterministic side effects and lack state-sensitive side effects). Pronouns will denote *non-constant* functions from stacks into *singleton* sets of values, as in Definition 2.20 (i.e. pronouns have only state-sensitive side effects and lack nondeterministic side effects).

Definition 2.19 (Indefinites in the Reader.Set monad).

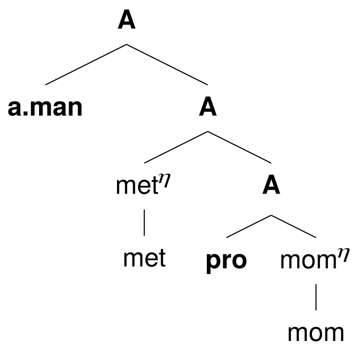
$$\begin{aligned} [\![\text{a man}]\!] &:= \textbf{a.man} \\ &:= \lambda s.\, \{x \mid \mathsf{man}\, x\} \end{aligned}$$

Definition 2.20 (Pronouns in the Reader.Set monad).

$$\begin{aligned} [\![\text{s/he}]\!] &:= \textbf{pro} \\ &:= \lambda s.\, \{s_\top\} \end{aligned}$$

Let's look at a simple example that exploits both the state-sensitive and nondeterministic aspects of the Reader.Set monad, namely *a man met her mom*. We lift the value-denoting *met* and *mom* into the monad via applications of η, interpret binary-branching nodes with **A**, Rebracket, and simplify via LeftID, η, and $(\multimap)$. As expected, we end up with a non-constant function from states to sets of booleans, type $\mathsf{M}t$. The nondeterminism engendered by **a.man** and the state-sensitivity engendered by **pro** are both inherited by the larger program denoted by the whole sentence. The resulting program π is *true* at some s iff $\mathsf{True} \in \pi\, s$, i.e. iff there is some man who met $s_\top$'s mom.

(2.5) $[\![\text{a man met}^\eta \text{ her mom}^\eta]\!] =$



$$\begin{aligned}
&= \textbf{a.man}_x \multimap \mathsf{met}^\eta_f \multimap \textbf{pro}_y \multimap \mathsf{mom}^\eta_g \multimap (f\ (g\ y)\ x)^\eta && \text{Rebracket} \\
&= \textbf{a.man}_x \multimap \textbf{pro}_y \multimap (\mathsf{met}\,(\mathsf{mom}\, y)\, x)^\eta && \text{LeftID} \\
&= \textbf{a.man}_x \multimap \textbf{pro}_y \multimap \lambda s.\, \{\mathsf{met}\,(\mathsf{mom}\, y)\, x\} && \eta \\
&= \textbf{a.man}_x \multimap \lambda s.\, \{\mathsf{met}\,(\mathsf{mom}\, s_\top)\, x\} && \multimap \\
&= \lambda s.\, \{\mathsf{met}\,(\mathsf{mom}\, s_\top)\, x \mid \mathsf{man}\, x\} && \multimap
\end{aligned}$$

Before concluding, I'd like to flag a point on binding that will be relevant later. Though the

Reader.Set monad, like the Reader monad, traffics in state-sensitive effects and can handle unbound readings of pronouns (as we've just seen), it has no correspondent of the Reader monad's "Derived VP" operator $\lhd$. This is due to a very basic conflict: the output of the Derived VP rule must have the type $\mathsf{M}(e \to t)$; in the Reader.Set monad, this means it must be of the form $\lambda s.\, \{\lambda a.\, \cdots\}$; however, the only way to define an operation with the right effect—namely, that the state at which the VP is evaluated is parametrized to the VP's argument—is to have the result be of the form $\lambda sa.\, \{\cdots\}$, which can't be a Reader.Set program, let alone the meaning of a Reader.Set VP (see Shan 2004 for a related point about the impossibility of defining a predicate abstraction operation in Kratzer & Shimoyama's 2002 SSNA semantics). We'll return to this problem and the Reader.Set monad in Chapter 5.

2.3.5 Wrapup

The Reader monad corresponds to the way variable binding is generally analyzed in static semantics. The Set monad gives pointwise functional application. And the Reader.Set monad does both state-sensitivity and nondeterminism at once. What I'd like to suggest is that we only need to make a very small departure from the Reader.Set monad to get to a robust theory of dynamic anaphora, one which I'll put to work in later chapters. We need only replace the Reader monad with something more general—i.e. a monad that allows for *state-changing* side effects (output) in addition to mere dependencies on the state (input)—and then find a way to combine this with the Set monad. In fact, once we have a monad for state-changing side effects (the State monad) in hand, the way forward is determined. We combine the State and Set monads in precisely the same way we combined the Reader and Set monads (i.e. using a monad transformer). The result is a robust theory of dynamic semantics in terms of side effects that informs the analyses in subsequent chapters.

2.4 Crash course in dynamic semantics and DyS

One of my main claims is that rethinking dynamic semantics in terms of side effects (and thus in terms of monads) is conceptually and empirically fruitful. So, bracketing monads for a few pages, let's learn a bit about dynamic semantics before we see how to apply the monadic perspective to it. Presupposing no prior knowledge of dynamic semantics, this section introduces some of the data motivating dynamic approaches to anaphora and gives a bare-bones (and novel) dynamic semantics for anaphora called DyS. The key properties of DyS are the following. First, binding without scope is achieved by adopting a richer notion of meaning that allows sentences to *output* possibly updated states. Second, *nondeterminism* allows indefinites to be treated as nondeterministic referring expressions, i.e. as the nondeterministic analogs of proper names.

2.4.1 Empirical motivation

Syntactically singular DPs can fix the reference of ("bind") singular pronouns, see (2.6). The textbook treatment of binding requires the binder to *take scope* over a meaning which covalues the argument slots corresponding to binder and bind-ee. See (2.7), where Q ranges over quantifier meanings and the argument of head is covalued with the external argument of rubbed. Call this *in-scope binding*. Assuming for illustration that Q ranges over extensional generalized quantifier meanings, type $(e \to t) \to t$ (e.g. $[\![\text{exactly one linguist}]\!] = \lambda k.\, \exists! x.\, \mathsf{ling}\, x \wedge k\, x$), in-scope binding assigns adequate truth conditions to the sentences in (2.6) (e.g. $\exists! x.\, \mathsf{ling}\, x \wedge \mathsf{rubbed}\, (\mathsf{head}\, x)\, x$).

(2.6)
$$\left.\begin{array}{l}\text{Polly}_i \\ \text{A linguist}_i \\ \text{Exactly one linguist}_i \\ \text{Every linguist}_i \\ \text{No linguist}_i\end{array}\right\} \text{rubbed her}_i \text{ head.}$$

(2.7)
$$Q\, (\lambda x.\, \mathsf{rubbed}\, (\mathsf{head}\, x)\, x)$$

Though simple and powerful, this picture is hard to square with other data. It turns out that *Polly*, *a linguist*, and *exactly one linguist* can all seemingly bind a singular pronoun in a separate clause, see (2.8). But this apparent binding relationship is off limits for *every linguist* and *no linguist*. Structure seems to foreclose binding relationships for some DPs but not for others. Though this is not so troublesome for the *Polly* case (the "binding" could just be a bit of serendipitous "accidental" coreference), the matter is more pressing for *a linguist* and *exactly one linguist*, which do not seem to be expressions which could co-refer with the pronoun.

(2.8)
$$\left.\begin{array}{l}\ \ \text{Polly}_i \\ \ \ \text{A linguist}_i \\ \ \ \text{Exactly one linguist}_i \\ {}^*\text{Every linguist}_i \\ {}^*\text{No linguist}_i\end{array}\right\} \text{yawned. She}_i \text{ was tired.}$$

(2.9)
$$Q\, (\lambda x.\, \mathsf{yawn}\, x \wedge \mathsf{tired}\, x)$$

There's reason to think that the successful cases in (2.8) do not involve in-scope binding, as in (2.9). For one, the requisite scoping—out of a tensed clause and to a position of *text-level* scope—is implausible, especially so in theories which do scope-taking with syntactic transformations. Relatedly, a scoping account is silent about why it should be that *a/exactly one linguist* can take scope in this way, but *every/no linguist* cannot. And, fatally, giving *exactly one linguist* scope over the pronoun in this way simply yields *incorrect truth conditions*, requiring there to be exactly one linguist who yawned and was tired, i.e. $\exists! x.\, \mathsf{linguist}\, x \wedge \mathsf{yawn}\, x \wedge \mathsf{tired}\, x$. Yet this truth condition can satisfied if *many* linguists yawned, so long as just one of the yawning linguists was tired. What we want, instead, are truth conditions requiring that exactly one linguist yawned, and *that linguist* was tired. That freely scoping *exactly one linguist* over the pronoun gives an unattested interpretation rather than this one strongly suggests that there is a grammatical distinction between in-scope and out-of-scope binding, and that we need some apparatus for doing the latter.

2.4.2 Discourse referents

A basic tenet of anaphoric dynamic semantics is that interpretation happens relative to a discourse, and processing a sentence can induce changes in a discourse which facilitate the processing of pronouns downstream. Drawing on terminology introduced in Karttunen 1976, we might put it this way: sentences are in the business of creating *discourse referents* ('drefs', for short), and pronouns are in the business of retrieving them. As an example, a name can bind a pronoun it doesn't scope over because processing the sentence in which the name occurs has the effect of adding a dref for the individual named by the name (which the downstream pronoun can access). Indefinites must work analogously, and quantifiers like *every/no/most linguist(s)* must somehow be different.

This is a richer picture than one which identifies sentence meanings with truth conditions, and some new machinery is needed needed to get things going. Most pressingly for our purposes, we'll need to get a handle on what it means to introduce a dref, as well as what sorts of drefs get introduced by indefinites. This section sketches a minimal dynamic interpretation scheme for making and accessing drefs called DyS. The system's closest relative is Dekker's 1994 Predicate Logic with Anaphora, though the essential features of DyS (in particular, the idea that sentences create drefs, and that indefinites are the nondeterministic analogs of proper names) characterize essentially all dynamic interpretation schemes.[11]

We begin by fixing a data structure for the state, i.e. where drefs live. The requirements are quite minimal; something as basic as a set could do the job (cf. de Groote 2006)—or indeed a simple functional dependency on individuals (e.g. Jacobson 1999, Shan 2001, Szabolcsi 2003, Shan & Barker 2006). Practically speaking, however, it's easier to have something with more structure. We'll use linear sequences of drefs, aka *stacks*. I'll write stacks as strings of values, like so: $\omega ... cba$. Variables ranging over stacks will be s, s', and s'', and the type of stacks is also given by the homonym s.

Stacks can be extended with drefs. I'll write $\widehat{sx}$ for the stack s extended with the dref x; e.g. if s is jb then $\widehat{s\mathsf{m}}$ is jbm. They can also be examined for drefs. In what follows, we'll (perhaps surprisingly) only ever need to retrieve the last element of a stack s, which I'll write $s_\top$ for "topical", see Definition 2.21. The rough idea is that given some stack s, s's last member is the most recently introduced dref, its next-to-last member the next most recent, and so on (this is an extremely crude measure of topicality, but it'll be sufficient for our purposes; see e.g. Grosz et al. 1995, Hardt 1999, Bittner 2001, Stojnic et al. 2013, Bittner 2014 for more on topicality vis à vis anaphora). Notice that for any s and a, $\widehat{sa}_\top = a$. We'll see this equivalence a bunch in what follows.

11 See e.g. Heim 1982, Kamp 1981, Barwise 1987, Rooth 1987, Groenendijk & Stokhof 1991, Dekker 1994, van den Berg 1996, Muskens 1996, Bittner 2001, Brasoveanu 2007, 2008. A different strategy is pursued in Groenendijk & Stokhof 1990, Szabolcsi 2003, de Groote 2006, where sentences give information about their *possible continuations*, rather than about the state of the discourse per se. Like Dekker, DyS uses *sequences of drefs* constructed on the fly in the course of interpretation to model discourse reference. Unlike Dekker, DyS makes no use of assignment functions and is used to interpret a fragment of English rather than a logical language.

Definition 2.21 (Topical dref retrieval).

$$s_{\top} := \text{the last element of } s$$

A sentence in DyS denotes a function from an input stack into a set of zero or more output stacks. In other words, DyS sentence meanings are 2-place *relations on stacks*, type $s \to s \to t$. (A relation between stacks is isomorphic to a function from stacks into sets of stacks, and I'll freely move between relation and function-into-sets talk in what follows.) We'll abbreviate type $s \to s \to t$ as $\mathbf{t}$, and refer to things of type $\mathbf{t}$ as *dynamic propositions*.

Here's a simple example of dynamic propositions and the input-output perspective at work. Assuming Polly yawned, processing *Polly yawned* takes us from an input stack s to an output $\widehat{s\mathsf{p}}$ (here I characterize the relation denoted by the sentence using a more iconic infix notation). Notice how the normal truth conditions of this sentence, viz. that Polly yawned, turn into a condition on the relation being nonempty: given an s, there's an s' such that s ⟦Polly yawned⟧ s' iff Polly yawned.

(2.10) $$s\ [\![\text{Polly yawned}]\!]\ s' \text{ iff } \mathsf{yawn}\,\mathsf{p} \wedge s' = \widehat{s\mathsf{p}}$$

A parallel example with an indefinite is given in (2.11). On the righthand side of the biconditional, p has been replaced with a variable bound by an existential quantifier restricted to linguists. No big surprise there, but in the relational setup this has an interesting upshot (familiar to anyone who's seen Karttunen's 1977 semantics for interrogatives): (2.11) potentially outputs a *multitude* of stacks. Suppose the yawning linguists are p, q, and r. Given an input stack s, the outputs are $\{\widehat{s\mathsf{p}}, \widehat{s\mathsf{q}}, \widehat{s\mathsf{r}}\}$. Again, the truth conditions turn into a condition on there being outputs: ⟦a linguist yawned⟧ is nonempty iff there's a linguist who yawned.

(2.11) $$s\ [\![\text{a linguist yawned}]\!]\ s' \text{ iff } \exists x.\, \mathsf{ling}\, x \wedge \mathsf{yawn}\, x \wedge s' = \widehat{sx}$$

We'll say that processing *a linguist yawned* has the effect of extending an input stack with a *random* or *nondeterministic* linguist who yawned, with randomness/nondeterminism cashed out in terms of multiple outputs. Thus, indefinites are like proper names, but differ in that they potentially trigger a nondeterministic update. Accommodating nondeterminism is in the end the reason for DyS to type propositions as *relations* on stacks (i.e. functions from stacks into sets of stacks).

2.4.3 Compositionally

DyS treats the meanings of VPs as *dynamic properties*, i.e. functions from individuals into dynamic propositions (type $e \to \mathbf{t}$). For example, the meaning of the intransitive predicate *yawned*, i.e. ⟦yawned⟧, takes an individual x and an input stack s and outputs s (and nothing else) iff x yawned. Meanings like ⟦yawned⟧ are *tests*: they only check to make sure that their arguments satisfy some condition. If the test is passed, the input stack is returned unchanged. If the test fails, the output is empty. (NB: not all dynamic properties are tests!)

Definition 2.22 (A DyS verb phrase meaning).

$$[\![\text{yawned}]\!] \coloneqq \lambda x s.\, \{s\} \text{ if } \mathsf{yawn}\, x \text{ else } \{\,\}$$

DyS DPs denote functions from dynamic properties to dynamic propositions, type $(e \to \mathbf{t}) \to \mathbf{t}$. See Definition 2.23. Proper names feed a dynamic property k Polly and a stack extended with Polly. Pronouns retrieve the topical dref from a stack, feeds it to some scope k, and leave the stack unchanged. And indefinites like *a linguist* feed k a *random* individual and a stack extended with that random individual. For example, if the linguists are o, p, q, and r, $[\![\text{a linguist}]\!]$ is shorthand for $\lambda k s.\, k\,\mathsf{o}\,\widehat{s\mathsf{o}} \cup k\,\mathsf{p}\,\widehat{s\mathsf{p}} \cup k\,\mathsf{q}\,\widehat{s\mathsf{q}} \cup k\,\mathsf{r}\,\widehat{s\mathsf{r}}$. The idea is that this corresponds to a refusal to choose between the possible referents; it's up to the discourse to eventually narrow things down (or not). DyS indefinites are thus nondeterministic, *non-quantificational* analogs of proper names.[12]

Definition 2.23 (DyS DPs).

$$\begin{aligned} [\![\text{Polly}]\!] &\coloneqq \lambda k s.\, k\,\mathsf{p}\,\widehat{s\mathsf{p}} \\ [\![\text{a linguist}]\!] &\coloneqq \lambda k s. \bigcup_{\mathsf{ling}\, x} k\,x\,\widehat{sx} \\ [\![\text{she}]\!] &\coloneqq \lambda k s.\, k\, s_\top\, s \end{aligned}$$

Since DyS DPs have the type $(e \to \mathbf{t}) \to \mathbf{t}$, and VPs have the type $e \to \mathbf{t}$, the meanings of sentences like *Polly/she/a linguist yawned* can be composed by simple functional application. In the first case, we output a single stack extended with a dref for Polly (so long as Polly yawned). In the latter case, we output a set of stacks, each extended with a linguist who yawned. These are precisely the input-output conditions characterized in (2.10) and (2.11). See Fact 2.6, where I include an additional example with a pronoun to drive the point home.

Fact 2.6 (DyS sentences).

$$\begin{aligned} [\![\text{Polly yawned}]\!] &= [\![\text{Polly}]\!]\, [\![\text{yawned}]\!] \\ &= \lambda s.\, \{\widehat{s\mathsf{p}}\} \text{ if } \mathsf{yawn}\,\mathsf{p} \text{ else } \{\,\} \\ [\![\text{a linguist yawned}]\!] &= [\![\text{a linguist}]\!]\, [\![\text{yawned}]\!] \\ &= \lambda s.\, \{\widehat{sx} \mid \mathsf{ling}\, x \wedge \mathsf{yawn}\, x\} \\ [\![\text{she yawned}]\!] &= [\![\text{she}]\!]\, [\![\text{yawned}]\!] \\ &= \lambda s.\, \{s\} \text{ if } \mathsf{yawn}\, s_\top \text{ else } \{\,\} \end{aligned}$$

Sentences with proper names or indefinites output drefs, and sentences with pronouns access a dref and use it to fix their reference. This means that securing dynamic binding should be a fairly simple matter of getting the plumbing right, such that the stacks output by one sentence manage to

12 This point may be over-stated: Karttunen 1977 shows that question words like *who* can give rise to relational meanings even though their semantics is very much quantificational. Notice in this respect that there is a *bona fide* existential quantifier on the righthand side of the biconditional in (2.11). I won't emphasize this, but it's worth keeping in mind that Karttunen's semantics for interrogatives may have something to teach us about the dynamic treatment of indefinites.

get shepherded to the pronouns in another. Along these lines, conjunction is defined as relational composition. It turns two sentences into a stack assembly line: the stacks output by the first conjunct are fed, one by one, to the second, and the resulting outputs are collected.

Definition 2.24 (DyS conjunction).

$$[\![;]\!] \coloneqq \lambda r l s. \bigcup_{s' \in l s} r s'$$

While binding from the left conjunct into the right conjunct is effortless, binding from the right conjunct into the left conjunct is unacceptable, see (2.12). Relational composition is asymmetric in a way consistent with these facts: the stacks output by the left conjunct are fed to the right conjunct, and not vice versa.

(2.12) a. A linguist$_i$ yawned; she$_i$ was tired.
b. *She$_i$ was tired; a linguist$_i$ yawned.

A couple examples of how conjunction enables anaphora from one clause into another are given in Fact 2.7. In both of these cases, the semantics of DyS conjunction funnels drefs (qua modified stack) from the first conjunct to the second. The second-line simplifications are due to the meaning of $\top$. The deterministic case outputs a stack extended with p so long as Polly yawned and was tired. The nondeterministic case outputs a stack extended with an x so long as x is a linguist who yawned and was tired. Binding succeeds for both.

Fact 2.7 (DyS sequencing).

$$\begin{aligned} s\,[\![\text{Polly yawned; she was tired}]\!]\,s' &\text{ iff } \mathsf{yawn}\,\mathsf{p} \wedge \mathsf{tired}\,s'_{\top} \wedge s' = \widehat{s\mathsf{p}} \\ &\text{ iff } \mathsf{yawn}\,\mathsf{p} \wedge \mathsf{tired}\,\mathsf{p} \wedge s' = \widehat{s\mathsf{p}} \\ s\,[\![\text{a linguist yawned; she was tired}]\!]\,s' &\text{ iff } \exists x.\,\mathsf{ling}\,x \wedge \mathsf{yawn}\,x \wedge \mathsf{tired}\,s'_{\top} \wedge s' = \widehat{sx} \\ &\text{ iff } \exists x.\,\mathsf{ling}\,x \wedge \mathsf{yawn}\,x \wedge \mathsf{tired}\,x \wedge s' = \widehat{sx} \end{aligned}$$

In addition to allowing anaphora across clausal boundaries, this semantics gives an adequate rendering of the truth conditions in each case. The deterministic text outputs a stack iff Polly yawned and was tired. The nondeterministic text outputs a stack iff there is a yawning, tired linguist.

2.4.4 Wrapup

The purpose of this section was to illustrate a dynamic interpretation scheme and to draw out the two key properties which we will model as side effects in the following section and beyond: the notion of outputting a modified input, and a nondeterministic semantics for indefinites. We did not cover other sorts of DPs (e.g. plural indefinites or DPs headed by *every, no, exactly one*). Nor have we said anything about scope-taking, quantification, scope inversion, and so on. We'll return to these issues in later chapters.

I briefly note some interesting features of DyS. We have a full dynamic semantics (and one that can be extended in the usual ways), but we have not availed ourselves of co-indexation between antecedents and pronouns (van Eijck 2001); coreference and binding are not syntactically represented in DyS (see e.g. Fiengo & May 1994, Reinhart 2006, Safir 2004). Instead, a reference stack is constructed on the fly in the course of interpretation as a record of the individuals under discussion, so the semantics is in a sense variable-free (Jacobson 1999). Moreover, since stack updates are always monotonic (i.e. we only have the option of adding binding information to a stack, and never of deleting it), there is no possibility of an update destroying information as in standard assignment-based dynamic semantics.

2.5 Monadic dynamics

DyS is all about modifying the state and nondeterminism. But we already *have* a monad for nondeterminism: the Set monad! This suggests that all we need to do to arrive at a monadic grammar for the side effects in DyS is find a monad for changing the state (i.e. updating and outputting stacks) and then find a hybrid of this monad and the Set monad—exactly as we did with the Reader.Set monad. This will mean finding a monad for updating stacks, finding the corresponding monad transformer, and then applying the monad transformer to the Set monad. This is exactly how we'll proceed.

2.5.1 Output: the State monad

DyS secures discourse anaphora by supposing that part of what sentences do is to *output* modified input stacks. In DyS, this allows a sentence to actually *host* discourse-level information. There is no such possibility in the Reader monad. The Reader monad, though it gets input stacks and, in the presence of $\triangleleft$, potentially modifies them, affords no ability to *output* stacks. Reader-monadic programs only encode information about the ways the value they return does (or does not) depend on the state, and so they only tell half the story. We need a way to output modified stacks rather than tossing them on evaluation. We need, in short, a way to *change the state*.

The monad for doing this is the State monad. A State-monadic program is a function from a stack to a *pair* of a value and a possibly-updated stack. Thus a monadic individual, type $\mathsf{M}e$, is a function from a stack to a pair of an individual and a stack; a monadic boolean, type $\mathsf{M}t$ is a function from a stack to a pair of a boolean and a stack. The injection function just takes a value a and returns a function from a stack s to the pair $\langle a,\ s\rangle$. Lastly, sequencing m with $\lambda v.\,\pi$ means feeding m a stack to expose a pair $\langle a,\ s'\rangle$ of a value a and a possibly updated stack s', binding v to a in π, and then passing $\pi[a/v]$ the possibly updated stack s'.[13]

13 Two-dimensionality is actually not an essential part of the formalization. The "Church encoding" (cf. Church 1936) of ordered pairs treats them as higher-order functions. Thus $\langle a, b\rangle := \lambda k.\, k\, a\, b$, and $\alpha \times \beta := (\alpha \to \beta \to \rho) \to \rho$ (fixing some "result type" ρ). This means sequencing in the State monad can be given as $m \multimap k = \lambda s.\, m\, s\, k$. What's important for us is that values and stacks are both useful things to have and that neither should be discarded on evaluation. Pairs offer a convenient and intuitive notation for this idea, but no more than that.

Definition 2.25 (The State monad).

$$\begin{aligned}
\mathsf{M}\alpha &::= s \to (\alpha \times s) \\
a^{\eta} &:= \lambda s.\, \langle a,\ s \rangle \\
m_v \multimap \pi &:= \lambda s.\, \pi[a/v]\, s' \\
&\qquad \text{for } \langle a, s' \rangle := m\, s
\end{aligned}$$

In the State monad, the side effect is a parameter that gets threaded and updated as programs are evaluated. This differs from the Reader monad, and allows it to be the case that the incoming state is no longer *shared*; the state passed to $\pi[a/v]$ is a possibly *updated* stack.[14]

The State monad determines a new notion of monadic application. Monadic application of two State-monadic programs m and n means first evaluating m at some input stack to expose a value x and a possibly updated stack s', evaluating n at s' to expose a value y and a possible updated stack s'', and finally returning a pair of $\mathsf{A}\, x\, y$ and s''. See Figure 2.1 for an illustration, and notice how the stack is threaded from one program to the next. This corresponds in an especially clear way to the notion of a side effect; the stack is quite literally on the side! (Notice that the directionality of the threading matters; the State monad is the first monad we've seen that's not commutative, where order of evaluation actually matters.)

Fact 2.8 (Application in the State monad).

$$\begin{aligned}
\mathbf{A}\, m\, n = \lambda s.\, &\langle \mathsf{A}\, x\, y,\ s'' \rangle \\
&\text{for } \langle x,\ s' \rangle := m\, s \\
&\text{for } \langle y,\ s'' \rangle := n\, s'
\end{aligned}$$

Proof.

$$\begin{aligned}
\mathbf{A}\, m\, n &= m_v \multimap n_u \multimap (\mathsf{A}\, v\, u)^{\eta} && \mathbf{A} \\
&= m_v \multimap n_u \multimap \lambda s''.\, \langle \mathsf{A}\, v\, u,\ s'' \rangle && \eta \\
&= m_v \multimap \lambda s'.\, \langle \mathsf{A}\, v\, y,\ s'' \rangle && \multimap \\
&\qquad\qquad \text{for } \langle y,\ s'' \rangle := n\, s' \\
&= \lambda s.\, \langle \mathsf{A}\, x\, y,\ s'' \rangle && \multimap \\
&\quad \text{for } \langle x,\ s' \rangle := m\, s \\
&\quad \text{for } \langle y,\ s'' \rangle := n\, s'
\end{aligned}$$

□

The State monad represents a *generalization* of the Reader monad, in the following sense: if no programs ever update the stack, the functionality of the State monad reduces to that of the Reader

14 State monads have cropped up before in semantics. Giorgolo & Asudeh 2012 use a State monad to model conventional implicatures. Giorgolo & Unger 2009, Unger 2012 connect the State monad to anaphora. Of the two, Unger's 2012 use of State is most similar to mine, but her treatment of indefinites and quantification rely on representational properties of variable names in a way that makes her semantics distinct from mine and from standard dynamic theories.

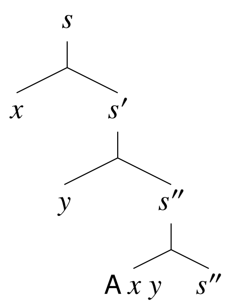


Figure 2.1: Visualizing application in the State monad.

monad. But State-monadic programs can do things Reader-monadic programs can't—they can *hold binding information*. In the Reader monad, a program could accept a state and transmute it into binding potential *for that program*. In the State monad, binding potential is liberated from the program that's being bound into.

The upshot is that there's a way to give constituents an anaphoric charge, as in Definition 2.26, which implies the corresponding rule of interpretation in Definition 2.27. Applying $\triangleright$ to a program m means creating a dref corresponding to the value returned by m. For example, for any value a, $a^{\eta\triangleright} = \lambda s.\, \langle a,\, \widehat{sa}\rangle$. Notice that Definition 2.26 is one $\widehat{sv}$ away from an instantiation of RightID (i.e. from having no effect): thus, the only effect of $\triangleright$ is the introduction of a dref.

Definition 2.26 (Dref introduction in the State monad).

$$\begin{aligned} m^{\triangleright} &:= m_v \multimap \lambda s.\, v^{\eta}\, \widehat{sv} \\ &= m_v \multimap \lambda s.\, \langle v,\, \widehat{sv}\rangle \end{aligned}$$

Definition 2.27 (Interpretation with $\triangleright$ in the State monad).

$$[\![X^{\triangleright}]\!] := [\![X]\!]^{\triangleright}$$

Sequenced programs annotated with $\triangleright$ can be simplified along the lines of Fact 2.9. Sequencing a bind-shifted program $m^{\triangleright}$ with $\lambda v.\, \pi$ is equivalent to sequencing m with $\lambda vs.\, \pi\, \widehat{sv}$.

Fact 2.9 (Simplifying $\triangleright$ programs in the State monad).

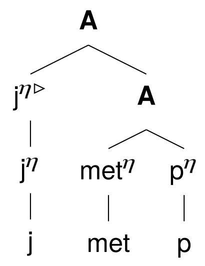
$$m^{\triangleright}_v \multimap \pi = m_v \multimap \lambda s.\, \pi\, \widehat{sv}$$

Proof. We begin with some routine simplifications:

$$\begin{aligned} m^{\triangleright}_v \multimap \pi &= (m_u \multimap \lambda s.\, \langle u,\, \widehat{su}\rangle)_v \multimap \pi && \text{Definition 2.26} \\ &= m_u \multimap (\lambda s.\, \langle u,\, \widehat{su}\rangle)_v \multimap \pi && \text{Assoc} \\ &= m_u \multimap \lambda s.\, \pi[u/v]\, \widehat{su} && \multimap \end{aligned}$$

Since the last line unifies u and the free occurrences of v in π, it's equivalent to $m_v \multimap \lambda s.\, \pi\, \widehat{sv}$. □

Let's go through an example to see how dref introduction works. We'll analyze the sentence *John met Polly*. We assume a standard right-branching parse, lifting everything into the monad via an application of η, and applying the $\triangleright$ operator to *John* (but not *Polly*). This pushes a discourse referent for John onto the stack. After Rebracketing and simplifying, we end up with a function from a stack s to a pair of a boolean that's True iff John met Polly, and s extended with j.

(2.13) $[\![\text{John}^{\eta\triangleright}\ \text{met}^{\eta}\ \text{Polly}^{\eta}]\!] =$ 

$$
\begin{aligned}
&= \mathsf{j}_x^{\eta\triangleright} \multimap \mathsf{met}_f^{\eta} \multimap \mathsf{p}_y^{\eta} \multimap (f\ y\ x)^{\eta} && \text{Rebracket}\\
&= \mathsf{j}_x^{\eta\triangleright} \multimap (\mathsf{met}\,\mathsf{p}\,x)^{\eta} && \text{LeftID}\\
&= \mathsf{j}_x^{\eta\triangleright} \multimap \lambda s.\,\langle \mathsf{met}\,\mathsf{p}\,x,\ s\rangle && \eta\\
&= \mathsf{j}_x^{\eta} \multimap \lambda s.\,\langle \mathsf{met}\,\mathsf{p}\,x,\ \widehat{sx}\rangle && \text{Fact 2.9}\\
&= \lambda s.\,\langle \mathsf{met}\,\mathsf{p}\,\mathsf{j},\ \widehat{s\mathsf{j}}\rangle && \text{LeftID}
\end{aligned}
$$

Had we replaced $Polly^{\eta}$ with $Polly^{\eta\triangleright}$, our output stacks would have the form $\widehat{s\mathsf{jp}}$ (i.e. s extended with j and then p). The linear evaluation bias in $[\![\cdot]\!]$ means that the order of drefs on the stack tracks the order the words introducing them occurred in the discourse.

The flip-side of stack modification is stack dependence, i.e. pronouns. Pronouns work about as you'd expect, returning a pair whose first member is the topical dref, and whose second value is the unchanged input stack. The only difference from the Reader monad is that the stack is passed along unmodified, as well. Since the only side-effect of the pronoun's semantics is its dependence on the stack, sequencing a pronoun with some computation π simply makes π dependent on the stack in the way the pronoun was.

Definition 2.28 (Pronouns in the State monad).

$$
\begin{aligned}
[\![\text{s/he}]\!] &:= \textbf{pro}\\
&:= \lambda s.\,\langle s_{\top},\ s\rangle
\end{aligned}
$$

Here is a handy rule for pronominal binding. Given that pronouns always pick the most recently introduced dref, whenever a pronoun finds itself within the immediate scope of a program θ to which $\triangleright$ has applied, we can be certain that the pronoun will evaluate to whatever θ evaluates to:

Fact 2.10 (Binding).

$$m_v^{\triangleright} \multimap \textbf{pro}_u \multimap \pi = m_v^{\triangleright} \multimap \pi[v/u]$$

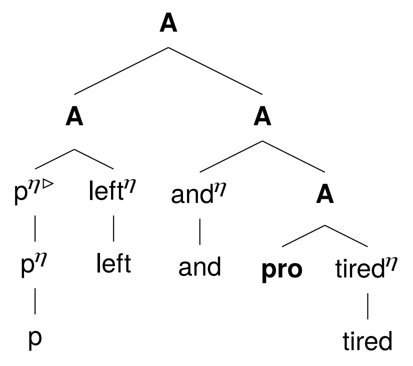
Proof.

$$\begin{aligned} m_\nu^\triangleright \multimap \textbf{pro}_u \multimap \pi &= m_\nu^\triangleright \multimap \lambda s.\, \pi[s_\top/u]\, s && \multimap \\ &= m_\nu \multimap \lambda s.\, \pi[\widehat{s\nu}_\top/u]\, \widehat{s\nu} && \triangleright \\ &= m_\nu \multimap \lambda s.\, \pi[\nu/u]\, \widehat{s\nu} && \top \\ &= m_\nu^\triangleright \multimap \pi[\nu/u] && \triangleright \end{aligned}$$

□

With State-monadic characterizations of dref introduction and the lexical semantics of pronouns, we have all we need to tackle a case of cross-sentential binding. All we need to assume is that the semantics of text sequencing is static conjunction(!), i.e. $[\![;]\!] = \mathsf{and} = \lambda qp.\, p \wedge q$. We lift everything but *she* into the monad via η, apply $\triangleright$ to *Polly*, and do monadic application. We Rebracket, use LeftID to reconstruct the trivial programs, and iteratively simplify. The contribution of **pro** is just to introduce a dependency on the stack, and the contribution of $\mathsf{p}^{\eta\triangleright}$ is to make a dref which fixes the referent of the pronoun. When all is said and done, we have a function from a stack s to a pair of a boolean (True iff Polly left and was tired) and s extended with Polly.

(2.14) $[\![\text{Polly}^{\eta\triangleright}\ \text{left}^{\eta}\ ;^{\eta}\ \text{she was-tired}^{\eta}]\!]$ =



$$\begin{aligned} &= \mathsf{p}_x^{\eta\triangleright} \multimap \mathsf{left}_f^{\eta} \multimap \mathsf{and}_g^{\eta} \multimap \textbf{pro}_y \multimap \mathsf{tired}_h^{\eta} \multimap (g\ (h\ y)\ (f\ x))^{\eta} && \text{Rebracket} \\ &= \mathsf{p}_x^{\eta\triangleright} \multimap \textbf{pro}_y \multimap (\mathsf{left}\, x \wedge \mathsf{tired}\, y)^{\eta} && \text{LeftID} \\ &= \mathsf{p}_x^{\eta\triangleright} \multimap (\mathsf{left}\, x \wedge \mathsf{tired}\, x)^{\eta} && \text{Binding} \\ &= \mathsf{p}_x^{\eta\triangleright} \multimap \lambda s.\, \langle \mathsf{left}\, x \wedge \mathsf{tired}\, x,\ s\rangle && \eta \\ &= \mathsf{p}_x^{\eta} \multimap \lambda s.\, \langle \mathsf{left}\, x \wedge \mathsf{tired}\, x,\ \widehat{sx}\rangle && \text{Fact 2.9} \\ &= \lambda s.\, \langle \mathsf{left}\, \mathsf{p} \wedge \mathsf{tired}\, \mathsf{p},\ \widehat{s\mathsf{p}}\rangle && \text{LeftID} \end{aligned}$$

A fortiori, the bound reading of *John's mom likes him* (i.e. without surface c-command) presents no problems. A dref for John can be introduced and then simply passed rightward.

Like DyS, the State monad allows stacks to be updated and output. But the way this is accomplished is different in several important ways. Dynamic composition extends sub-clausally, and where dynamic semantics has two conditions on output stacks—viz. that there are some only if the truth condition is met, and the output stacks have to look so-and-so—we have a *pair* of a truth condition and the output stack. Moreover, the State monad semantics has no need for lexical dynamic conjunction; cross-sentential binding is thus achieved with a minimum of lexical stipulations. All we've done is adopt the State monad and assume a State-monadic semantics for dref introduction and pronouns. Rebracket entails that hierarchy does not constrain binding, and the leftward bias DyS hard-codes into the semantics of *and* lives in the grammar; the left-biased interpretation function entails that the effects pipeline runs left to right and not vice versa.

2.5.2 Output nondeterministically: the State.Set monad

Dynamic semantics is state-changing plus nondeterminism. Our monad for state-changing (which facilitates binding without c-command) is the State monad. Our monad for nondeterminism (which facilitates a nondeterministic treatment of indefinites) is the Set monad. This means a monad with both State- and Set- like side effects will be a suitable choice for analyzing discourse anaphora along the lines of DyS, but in terms of side effects.

This monad is the State.Set monad. Programs in the State.Set monad are nondeterministic variants of programs in the State monad: $s \to (\alpha \times s)$ in the State monad's M is replaced with $s \to (\alpha \times s) \to t$. Injecting a value into the monad means giving back a function from a stack into a singleton set of a pair of a value and a stack. And sequencing means doing State-monadic sequencing a bunch of times and collecting the results. The State.Set monad differs from the State monad in that values can be nondeterministic; it differs from the Set monad in that a stack is passed around.[15]

Definition 2.29 (The State.Set monad).

$$\begin{aligned}
\mathsf{M}\alpha &::= s \to (\alpha \times s) \to t \\
a^{\eta} &:= \lambda s.\, \{\langle a,\ s\rangle\} \\
m_v \multimap \pi &:= \lambda s. \bigcup_{\langle a, s'\rangle \in m\, s} \pi[a/v]\, s'
\end{aligned}$$

Application works much as in the State monad, except for the fact that we now potentially output a multitude of pairs. As in the State monad, a stack is threaded to m, from m to n, and finally into the composite program, and functional application is performed on the two values so obtained. The only difference from the State monad is that this now happens in a *nondeterministic* fashion: instead of following one stack as it's threaded from m to n, we have the ability to follow a multitude of them. And instead of returning a single value, we return a nondeterministic value by performing nondeterministic functional application on the values that result from evaluating m and n.

Fact 2.11 (Application in the State.Set monad).

$$\begin{aligned}
\mathbf{A}\, m\, n = \lambda s.\, \{\, \langle \mathsf{A}\, x\, y,\ s''\rangle \mid \langle x,\ s'\,\rangle \in m\, s\ \wedge \\
\langle y,\ s''\rangle \in n\, s'\, \}
\end{aligned}$$

15 Two comments. (i) Pairs are again inessential (cf. fn. 13): $m \multimap k$ can be given as $\lambda s. \bigcup_{f \in m s} f\, k$. (ii) State.Set programs bear a resemblance to the *parametrized sets* of early theories of dynamic semantics (Barwise 1987, Rooth 1987).

Proof.

$$\begin{aligned}
\mathbf{A}\, m\, n &= m_v \multimap n_u \multimap (\mathsf{A}\, v\, u)^{\eta} && \mathbf{A} \\
&= m_v \multimap n_u \multimap \lambda s''.\, \{\, \langle \mathsf{A}\, v\, u,\ s'' \rangle \,\} && \eta \\
&= m_v \multimap \lambda s'.\, \{\, \langle \mathsf{A}\, v\, y,\ s'' \rangle \mid \langle y,\ s'' \rangle \in n\, s' \,\} && \multimap \\
&= \lambda s.\, \{\, \langle \mathsf{A}\, x\, y,\ s'' \rangle \mid \langle x,\ s' \rangle \in m\, s \wedge \langle y,\ s'' \rangle \in n\, s' \,\} && \multimap
\end{aligned}$$

□

As with the Reader.Set monad, the State.Set monad is the result of applying a monad transformer to the Set monad. Here the monad transformer is StateT, a function from a monad $\mathcal{M}_1 = \langle \mathsf{M}_1,\ \eta_1,\ \multimap_1 \rangle$ into a new monad $\mathcal{M}_2 = \langle \mathsf{M}_2,\ \eta_2,\ \multimap_2 \rangle$ that wraps State-monadic functionality around $\mathcal{M}_1$, as in Definition 2.30. Applying StateT to the Set monad yields the State.Set monad. (Parallel to ReaderT, applying StateT to the Identity monad yields the State monad.)

Definition 2.30 (The StateT monad transformer).

$$\begin{aligned}
\mathsf{M}_2\, \alpha &::= s \to \mathsf{M}_1\, (\alpha \times s) \\
a^{\eta_2} &:= \lambda s.\, \langle a,\ s \rangle^{\eta_1} \\
m_v \multimap_2 \pi &:= \lambda s.\, (m\, s)_{\langle a, s' \rangle} \multimap_1 \pi[a/v]\, s'
\end{aligned}$$

Dref introduction in the State.Set monad works analogously to dref introduction in the State monad. In fact, the *form* of the rules is completely unchanged; the only difference is in the underlying monadic operations. So for instance dref introduction still involves sequencing with a tweaked version of η, a $\widehat{sv}$ away from a RightID instantiation (Definition 2.31), the rule for interpretation is the same (Definition 2.32), and the simplification rule remains unchanged (Fact 2.12):

Definition 2.31 (Dref introduction in the State.Set monad).

$$\begin{aligned}
m^{\triangleright} &:= m_v \multimap \lambda s.\, v^{\eta}\, \widehat{sv} \\
&= m_v \multimap \lambda s.\, \{\langle v,\ \widehat{sv} \rangle\}
\end{aligned}$$

Definition 2.32 (Interpretation with ▷ in the State.Set monad).

$$[\![X^{\triangleright}]\!] := [\![X]\!]^{\triangleright}$$

Fact 2.12 (Simplifying ▷ programs in the State.Set monad).

$$m_v^{\triangleright} \multimap \pi = m_v \multimap \lambda s.\, \pi\, \widehat{sv}$$

Proof. As with Fact 2.9, we begin with some routine simplifications:

$$\begin{aligned}
m_v^{\triangleright} \multimap \pi &= (m_u \multimap \lambda s.\, \{\langle u,\ \widehat{su} \rangle\})_v \multimap \pi && \text{Definition 2.26} \\
&= m_u \multimap (\lambda s.\, \{\langle u,\ \widehat{su} \rangle\})_v \multimap \pi && \text{Assoc} \\
&= m_u \multimap \lambda s.\, \pi[u/v]\, \widehat{su} && \multimap
\end{aligned}$$

Since the last line unifies u and the free occurrences of v in π, it's equivalent to $m_v \multimap \lambda s.\, \pi\, \widehat{sv}$. $\square$

We now have the grammatical resources to extend the State monad's hierarchy-insensitive treatment of anaphora to indefinites. Our key lexical entries: indefinites and pronouns. As in DyS, indefinites are the nondeterministic analogs of proper names; whereas j^η is $\lambda s.\, \{\langle \mathsf{j},\, s\rangle\}$, a function into a singleton set, **a.man** is a function into a *multitude of pairs* (Definition 2.33). Since **a.man** has only nondeterministic side effects, given any π, $\textbf{a.man}_v \multimap \pi = \lambda s.\, \bigcup_{\mathsf{man}\, x} \pi[x/v]\, s$.

Definition 2.33 (Indefinites in the State.Set monad).

$$\begin{aligned} [\![\text{a man}]\!] &:= \textbf{a.man} \\ &:= \lambda s.\, \{\langle x,\, s\rangle \mid \mathsf{man}\, x\} \end{aligned}$$

Pronouns are as expected. Given some s, they give back (a singleton set containing) a pair whose first member is the most recently introduced dref in s, and whose second member is the unmolested input stack s (Definition 2.34). As with the State monad, for any π, $\textbf{pro}_v \multimap \pi = \lambda s.\, \pi[s_\top/v]\, s$.

Definition 2.34 (Pronouns in the State.Set monad).

$$\begin{aligned} [\![\text{s/he}]\!] &:= \textbf{pro} \\ &:= \lambda s.\, \{\langle s_\top,\, s\rangle\} \end{aligned}$$

As with the State monad, whenever a pronoun gets caught in the immediate scope of a binder, the pronoun can be simplified away, bound to the binder's value:

Fact 2.13 (Binding).

$$m_v^\rhd \multimap \textbf{pro}_u \multimap \pi = m_v^\rhd \multimap \pi[v/u]$$

Proof. Exactly as for the State monad, but we rely on Fact 2.12, the State.Set correlate of Fact 2.9.

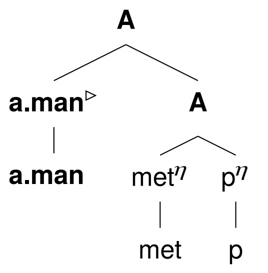
$$\begin{aligned} m_v^\rhd \multimap \textbf{pro}_u \multimap \pi &= m_v^\rhd \multimap \lambda s.\, \pi[s_\top/u]\, s && \multimap \\ &= m_v \multimap \lambda s.\, \pi[\widehat{sv}_\top/u]\, \widehat{sv} && \text{Fact 2.12} \\ &= m_v \multimap \lambda s.\, \pi[v/u]\, \widehat{sv} && \top \\ &= m_v^\rhd \multimap \pi[v/u] && \text{Fact 2.12} \end{aligned}$$

$\square$

Let's go through a basic example to get a sense of nondeterministic dref introduction. We'll analyze the sentence *a man met Polly*. We assume a standard right-branching parse, lifting the values *met* and *Polly* into the monad via an application of η and applying $\rhd$ to *a man*. After Rebracketing and simplifying, we end up with a function from a stack to a set of pairs. The result is the nondeterministic version of *John*$^{\eta\rhd}$ *met*$^\eta$ *Polly*$^\eta$—which the State monad interpreted as $\lambda s.\, \langle \mathsf{met}\, \mathsf{p}\, \mathsf{j},\, \widehat{s\mathsf{j}}\rangle$—where we replace the deterministic entity John with a nondeterministic man. (As before, had we replaced *Polly*$^\eta$

with *Polly*$^{\eta\rhd}$, our output stacks would have the form $\widehat{sx\mathsf{p}}$, with x a nondeterministic man.)

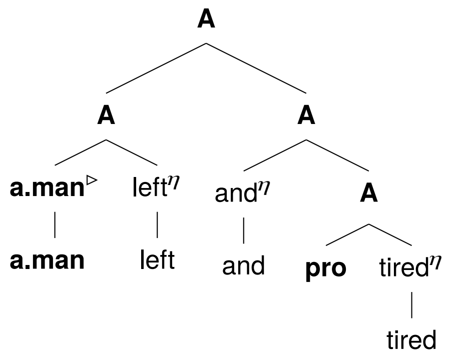
(2.15) $[\![\text{a man}^{\rhd}\ \text{met}^{\eta}\ \text{Polly}^{\eta}]\!]$ =



$$
\begin{aligned}
&= \textbf{a.man}^{\rhd}_{x} \multimap \mathsf{met}^{\eta}_{f} \multimap \mathsf{p}^{\eta}_{y} \multimap (f\ y\ x)^{\eta} && \text{Rebracket}\\
&= \textbf{a.man}^{\rhd}_{x} \multimap (\mathsf{met}\,\mathsf{p}\,x)^{\eta} && \text{LeftID}\\
&= \textbf{a.man}^{\rhd}_{x} \multimap \lambda s.\,\{\langle \mathsf{met}\,\mathsf{p}\,x,\ s\rangle\} && \eta\\
&= \textbf{a.man}_{x} \multimap \lambda s.\,\{\langle \mathsf{met}\,\mathsf{p}\,x,\ \widehat{sx}\rangle\} && \text{Fact 2.12}\\
&\lambda s.\,\{\langle \mathsf{met}\,\mathsf{p}\,x,\ \widehat{sx}\rangle \mid \mathsf{man}\,x\} && \multimap
\end{aligned}
$$

Here is the big payoff. With the State monad underlying output and the Set monad underlying nondeterminism, we can give a nondeterministic analysis of *a man left; he was tired* that parallels the State-monadic analysis of *Polly left; she was tired*. As with the State monad analog, we assume that the semantics of text sequencing is simply static conjunction. We give a standard parse to the text, lifting values into the monad and applying $\rhd$ to *a man*. After interpreting with monadic application, Rebracketing, and simplifying, we arrive at a nondeterministic variant of what we derived before.

(2.16) $[\![\text{a man}^{\rhd}\ \text{left}^{\eta}\ ;^{\eta}\ \text{he was tired}^{\eta}]\!]$ =



$$
\begin{aligned}
&= \textbf{a.man}^{\rhd}_{x} \multimap \mathsf{left}^{\eta}_{f} \multimap \mathsf{and}^{\eta}_{g} \multimap \textbf{pro}_{y} \multimap \mathsf{tired}^{\eta}_{h} \multimap (g\ (h\ y)\ (f\ x))^{\eta} && \text{Rebracket}\\
&= \textbf{a.man}^{\rhd}_{x} \multimap \textbf{pro}_{y} \multimap (\mathsf{left}\,x \wedge \mathsf{tired}\,y)^{\eta} && \text{LeftID}\\
&= \textbf{a.man}^{\rhd}_{x} \multimap (\mathsf{left}\,x \wedge \mathsf{tired}\,x)^{\eta} && \text{Binding}\\
&= \textbf{a.man}^{\rhd}_{x} \multimap \lambda s.\,\{\langle \mathsf{left}\,x \wedge \mathsf{tired}\,x,\ s\rangle\} && \eta\\
&= \textbf{a.man}_{x} \multimap \lambda s.\,\{\langle \mathsf{left}\,x \wedge \mathsf{tired}\,x,\ \widehat{sx}\rangle\} && \text{Fact 2.12}\\
&\lambda s.\,\{\langle \mathsf{left}\,x \wedge \mathsf{tired}\,x,\ \widehat{sx}\rangle \mid \mathsf{man}\,x\} && \multimap
\end{aligned}
$$

Notice that this last line is equivalent to $\textbf{a.man}_x^{\triangleright} \multimap (\textsf{left}\, x \wedge \textsf{tired}\, x)^{\eta}$, which cleanly separates the dynamic side effects from the static portions of the resulting meaning.

To recap, all we've done is merge the State and Set monads (via StateT applied to the Set monad). As with the State monad, binding is achieved without any special lexical entry for conjunction. Sequencing is a feature of the grammar and works no different between sentences from how it works inside of sentences. The linear bias DyS hard-codes into the semantics of *and* arises at the level of the grammar via the left-biased $[\![\cdot]\!]$. As with the Set monad, composition proceeds nondeterministically, enabling indefinites to work like proper names with indeterminate reference.

2.6 Wrapping up

This chapter explored monads as a way to explicitly incorporate side-effects—i.e. things which happen in the course of composition in addition to functional application—into semantic theorizing. We saw a few instances of ways that composition might be enriched to handle state-sensitivity, nondeterminism, and a mix, and we saw the monads to which these enrichments correspond.

After sketching a minimal dynamic interpretation system (DyS), I argued that dynamic semantics has two essential components: state (i.e. the ability to update and output discourse parameters) and nondeterminism (i.e. the ability to keep track of a number of computations at once). I proposed to treat these components as side effects. To do so, we found a monad for state (fittingly, State), the corresponding monad transformer (StateT), and applied the latter to the monad for nondeterminism (Set). The result was the State.Set monad, which we used to give a simple monadic grammar which incorporates dynamic side effects while retaining the basic shape of the grammars we used to talk about state-sensitivity, nondeterminism, and their combination (i.e. binary composition was still interpreted in terms of monadic application).

Intriguingly, our treatment of dynamic semantics in terms of side effects suggests a novel connection between dynamic semantics and static nondeterministic treatments of indefinites (e.g. Kratzer & Shimoyama 2002). Where the former relies on the State.Set monad, the latter relies on the Reader.Set monad. In other words, both recognize that indefinites have nondeterministic effects (reflected in the presence of the Set monad); where they differ is their approach to anaphora (reflected in the choice between Reader and Set). We'll explore this connection further in Chapter 5.

The next chapter extends this semantics into a full fragment with dynamically closed operators (i.e. negation and operators based on negation) and scope-taking. We'll introduce yet another notion of enriched composition, one that handles scopal side effects, via the *Continuation* monad. Far from forming an unwieldy whole, we'll see how the Continuation monad interfaces seamlessly with the apparatus developed in this chapter and how it offers an elegant and uniform perspective on meaning composition and the way side effects permeate their surroundings in the course of interpretation. In other words, we will see how side effects *take scope*.

Chapter 3

Scope and evaluation

3.1 Intro

This chapter extends the State.Set grammar motivated in Chapter 2 into a robust fragment with dynamically closed operators, quantifiers, and scope-taking. We begin by adding dynamically closed meanings (i.e. negation and meanings based on negation), including some that are irreducibly scopal (i.e. quantifiers like *every linguist* and *no linguist*). With scopal expressions in our quiver, we require a way to assemble programs more general than monadic application. To that end, I motivate and explore the Continuation monad for assembling programs with *scopal side effects*.

The intermediate result is a semantics that knows how to compose scopal programs that return *values*, but not ones that return *programs* (e.g. State.Set programs). The next move is to develop a general framework for scopal composition based on something I call an *underlying monad*, which allows for scopal programs that return programs in the State.Set monad—indeed, in *any* monad. The resulting grammar—consistent with any underlying regime for side effects—is to the value-returning Continuation-monadic grammar as the extended grammars were to a purely applicative grammar in Chapter 2. After fixing an underlying State.Set monad, which we converged on in Chapter 2 as the suitable choice for dynamic binding (we'll see other underlying monads in Sections 5.4 and 5.5), I offer treatments of binding, inverse scope, and the composition of complex DPs.

This chapter is fairly technical, so here are a few words to explain why I think it's worth the trouble. Recall my main claim: exceptional scope is side effects taking scope after evaluation. We developed one crucial part of this, i.e. a theory of side effects, in Chapter 2. This chapter complements the theory of side effects with a theory of scope-taking that allows side effects *to take scope*, and thus forms the basis for the theory of exceptional scope and empirical advances reported in Part II.

3.2 Dynamically closed meanings and scope

3.2.1 Dynamically closed operators

There are well-known constraints on when indefinites can bind pronouns (e.g. Groenendijk & Stokhof 1991). A dref introduced by an indefinite within the scope of a negation is inaccessible to pronouns beyond the negation's scope. See (3.1). Something similar holds for quantificational DPs like *every phonologist* and *no phonologist*. See (3.2). In addition to preventing indefinites in their scope from binding pronouns outside their scope, *every phonologist* and *no phonologist* are themselves incapable of binding pronouns that they do not scope over (despite the possibility of in-scope binding in cases like *every phonologist*$_i$ *twisted his*$_i$ *mustache*). See (3.3).

(3.1) Polly didn't meet a semanticist$_i$. She$_i$ works on Binding Theory. (*$\neg > \exists$)

(3.2) {Every, no} phonologist saw a semanticist$_i$. She$_i$ works on Binding Theory. (*$\forall > \exists$)

(3.3) *{Every, no} linguist$_i$ yawned. She$_i$ was tired.

We will call operators of this sort, which wipe out any drefs generated within their scope, *dynamically closed*. This section works toward an analysis of dynamically closed meanings in terms of the State.Set monad.

As a first step, we'll give a semantics for a dynamically closed negation, which we'll use to build other dynamically closed meanings (the standard way to proceed in dynamic semantics). Let's begin by reasoning about the semantics of negation *in DyS*; then we'll import its essential features into the State.Set setting. In DyS, sentences denote relations between input stacks and output stacks. Because drefs introduced by an indefinite within the scope of a negation are inaccessible to pronouns outside the negation, we can conclude that a negation takes a DyS proposition p and a stack s, checks that p is empty at s (i.e. false), and returns s unchanged if that turns out to be the case (and fails otherwise). See Definition 3.1 and Fact 3.1 for a re-expression in input-output terms.

Definition 3.1 (DyS negation).

$$[\![\text{not}]\!] = \lambda ps.\,\{s\} \text{ if } p\,s = \{\,\} \text{ else } \{\,\}$$

Fact 3.1 (DyS negation as input-output).

$$s\,[\![\text{not } S]\!]\,s' \text{ iff } s = s' \wedge \neg\exists s''.\,s\,[\![S]\!]\,s''$$

We can import this account directly into the State.Set setting. See Definition 3.2. Negation denotes a function from dynamic propositions to dynamic propositions, type $\mathsf{M}t \to \mathsf{M}t$. Negating a State.Set program gives a function from an input stack into a pair of something very much like the DyS truth condition—the only difference being that we look inside $m\,s$ for *pairs* of booleans and stacks rather than mere stacks—and an unchanged input stack. In general, the State.Set correlate of any dynamically closed DyS meaning outputting an input stack contingent on some truth condition will be a pair of the truth condition and the input stack; instead of treating boolean values as *conditions on* output, we treat them as proper values, i.e. as a bonafide component of the output.

Definition 3.2 (Negation).

$$\textbf{not} := \lambda ms.\,\{\langle \neg\exists s'.\,\langle \mathsf{True},\ s'\rangle \in m\,s,\ s\rangle\}$$

An example of how negation interacts with nondeterminism and drefs is given in Fact 3.2. We give a meaning for *it's false that a linguist left*. Though the indefinite is nondeterministic and, by virtue of $\rhd$, hosts a dref, the effect of the negation is to wipe out both. Here, the negation simply checks that there are no True booleans of the form left x, for x ranging over linguists. If that is so, it gives us a trivial program that returns True; otherwise, it gives us a trivial program that returns False. In either case, the nondeterminism and dref introduction associated with **a.ling**$^{\rhd}$ both disappear. The result is equivalent to a pure program.

Fact 3.2 (It's false that a linguist left).

$$\textbf{not}\,(\textbf{a.ling}^{\rhd}_{x} \multimap (\mathsf{left}\,x)^{\eta}) = (\neg\exists x.\, \mathsf{ling}\,x \wedge \mathsf{left}\,x)^{\eta}$$

Proof.

$$\begin{aligned}
\textbf{not}\,(\textbf{a.ling}^{\rhd}_{x} \multimap (\mathsf{left}\,x)^{\eta}) &= \textbf{not}\,(\lambda s.\, \{\langle \mathsf{left}\,x,\ \widehat{sx}\rangle \mid \mathsf{ling}\,x\}) && \textbf{a.ling},\ \rhd,\ \multimap,\ \eta \\
&= \lambda s.\, \{\langle \neg\exists s'.\, \langle \mathsf{True}\,s'\rangle \in \{\langle \mathsf{left}\,x,\ \widehat{sx}\rangle \mid \mathsf{ling}\,x\},\ s\rangle\} && \textbf{not} \\
&= \lambda s.\, \{\langle \neg\exists x.\, \mathsf{ling}\,x \wedge \mathsf{left}\,x,\ s\rangle\} && = \\
&= (\neg\exists x.\, \mathsf{ling}\,x \wedge \mathsf{left}\,x)^{\eta} && \eta
\end{aligned}$$

□

Notice that it isn't the case that **not** wipes out *every* side effect in its scope. In particular, if an unbound pronoun occurs within the scope of a negation, the semantics is equivalent to scoping a pronoun over the negation:

Fact 3.3 (Pronouns and negation).

$$\textbf{not}\,(\textbf{pro}_{v} \multimap \pi) = \textbf{pro}_{v} \multimap \textbf{not}\,\pi$$

Proof.

$$\begin{aligned}
\textbf{not}\,(\textbf{pro}_{v} \multimap \pi) &= \textbf{not}\,(\lambda s.\, \pi[s_{\top}/v]) && \textbf{pro},\ \multimap \\
&= \lambda s.\, \{\langle \neg\exists s'.\, \langle \mathsf{True},\ s'\rangle \in \pi[s_{\top}/v]\,s,\ s\rangle\} && \textbf{not} \\
&= \textbf{pro}_{v} \multimap \lambda s.\, \{\langle \neg\exists s'.\, \langle \mathsf{True},\ s'\rangle \in \pi\,s,\ s\rangle\} && \textbf{pro},\ \multimap \\
&= \textbf{pro}_{v} \multimap \textbf{not}\,\pi && \textbf{not}
\end{aligned}$$

□

Thus, it only really makes sense to speak of an operator as dynamically closed relative to some side effect. The meaning of negation is dynamically closed relative to nondeterminism and dref introduction, but because **not** π transparently reflects the anaphoric sensitivity of π, **not** is not dynamically closed *simpliciter*. Nevertheless, **not**'s effects on the nondeterminism and drefs in the negated program π will be of central in importance in what follows (but not so much its effects on pronouns), and so I will continue to speak loosely of dynamic closure *simpliciter*.

As is standard in dynamic frameworks, we can use dynamic negation to build up other dynamically closed meanings. For example, a conditional operator is naturally defined by drawing on the standard equivalence $p \Rightarrow q$ iff $\neg(p \wedge \neg q)$, as in Definition 3.3. *If* denotes a function from two monadic booleans into a third, type $\mathsf{M}t \rightarrow \mathsf{M}t \rightarrow \mathsf{M}t$.

Definition 3.3 (Conditionals).

$$\textbf{if} \coloneqq \lambda mn.\, \textbf{not}\, (m_p \multimap (\textbf{not}\, n)_q \multimap (p \wedge q)^{\eta})$$

In Fact 3.4, I use this meaning for the conditional operator to derive a case of donkey anaphora, i.e. the bound reading of *if someone knocked, she ran* (assuming $\textbf{so} \coloneqq \lambda s.\, \{\langle x,\ s\rangle \mid x_e\}$). The result is again equivalent to a trivial program, one which returns True iff every knocker ran (reasonable truth conditions for the sentence). While the dref generated in the antecedent by $\textbf{so}^{\triangleright}$ is available for the evaluation of the consequent, the program that ultimately results is devoid of both nondeterministic and anaphoric side effects.

Fact 3.4 (If someone walked, she ran).

$$\textbf{if}\, (\textbf{so}_x^{\triangleright} \multimap (\mathsf{k}\, x))\, (\textbf{pro}_y \multimap (\mathsf{r}\, y)^{\eta}) = (\forall x.\, \mathsf{k}\, x \Rightarrow \mathsf{r}\, x)^{\eta}$$

Proof.

$$\begin{aligned}
\textbf{if}\, (\textbf{so}_x^{\triangleright} \multimap (\mathsf{w}\, x)^{\eta})\, (\textbf{pro}_y \multimap (\mathsf{r}\, y)^{\eta}) &= \textbf{not}\, ((\textbf{so}_x^{\triangleright} \multimap (\mathsf{w}\, x)^{\eta})_p \multimap (\textbf{not}\, (\textbf{pro}_y \multimap (\mathsf{r}\, y)^{\eta}))_q \multimap (p \wedge q)^{\eta}) && \textbf{if}\\
&= \textbf{not}\, ((\textbf{so}_x^{\triangleright} \multimap (\mathsf{w}\, x)^{\eta})_p \multimap (\textbf{pro}_y \multimap (\neg \mathsf{r}\, y)^{\eta})_q \multimap (p \wedge q)^{\eta}) && \textbf{not}\\
&= \textbf{not}\, (\textbf{so}_x^{\triangleright} \multimap (\mathsf{w}\, x)_p^{\eta} \multimap \textbf{pro}_y \multimap (\neg \mathsf{r}\, y)_q^{\eta} \multimap (p \wedge q)^{\eta}) && \text{Assoc}\\
&= \textbf{not}\, (\textbf{so}_x^{\triangleright} \multimap \textbf{pro}_y \multimap (\mathsf{w}\, x \wedge \neg \mathsf{r}\, y)^{\eta}) && \text{LeftID}\\
&= \textbf{not}\, (\textbf{so}_x^{\triangleright} \multimap (\mathsf{w}\, x \wedge \neg \mathsf{r}\, x)^{\eta}) && \text{Binding}\\
&= (\neg \exists x.\, \mathsf{w}\, x \wedge \neg \mathsf{r}\, x)^{\eta} && \textbf{not}\\
&= (\forall x.\, \mathsf{w}\, x \Rightarrow \mathsf{r}\, x)^{\eta} && =
\end{aligned}$$

□

The result here lacks anaphoric (as well as nondeterministic) effects, and so we correctly predict that drefs generated by indefinites inside a conditional are inaccessible to pronouns outside the conditional. That is, *if someone knocked, she ran* cannot be felicitously continued with *she was in a hurry*.

3.2.2 Scopal expressions

Now we turn to dynamically closed quantificational DPs. Meanings for *every linguist* and *no linguist* are given in Definition 3.4. Their semantics is given in terms of negation and the meaning of a State.Set indefinite—the meaning for *every linguist* relies on the standard first-order equivalence between e.g. *every linguist left* and *there's no linguist who failed to leave*. These meanings have type $(e \rightarrow \mathsf{M}t) \rightarrow \mathsf{M}t$; they're functions from *dynamic properties*, type $e \rightarrow \mathsf{M}t$, into boolean-returning State.Set programs, type $\mathsf{M}t$

Definition 3.4 (Meanings for *every linguist* and *no linguist*).

$$\begin{aligned}\textsf{\textbf{ev.ling}} &\coloneqq \lambda k.\, \textsf{\textbf{not}}\,(\textsf{\textbf{a.ling}}_x \multimap \textsf{\textbf{not}}\,(k\,x))\\ \textsf{\textbf{no.ling}} &\coloneqq \lambda k.\, \textsf{\textbf{not}}\,(\textsf{\textbf{a.ling}}_x \multimap k\,x)\end{aligned}$$

If we fully expand these meanings into λ-theoretic terms, we find functions into singleton pairs of a truth condition and a stack. For example, **ev.ling** is provably equivalent to a fully expanded term $\lambda k s.\, \{\langle \forall x.\, \textsf{ling}\, x \Rightarrow \exists s'.\, \langle \textsf{True},\, s'\rangle \in k\, x\, s,\ s\rangle\}$.

An analysis of *every linguist met a historian* using **ev.ling** is given in Fact 3.5 (of course, I haven't yet said anything about how this meaning is compositionally assembled). I have elected to translate the indefinite as a program that contributes a dref so that we can get a sense of the quantifier's effect on anaphoric potential, as well as nondeterminism. The result is a trivial M*t* program that returns True iff every linguist met some historian or other. As with *it's false that a linguist left* and *if someone knocked, she ran*, the nondeterminism and anaphoric potential that inhere in the dynamically closed operator's scope disappear beyond its scope.

Fact 3.5 (Every linguist met a historian).

$$\textsf{\textbf{ev.ling}}\,(\lambda x.\, \textsf{\textbf{a.hist}}^{\triangleright}_{y} \multimap (\textsf{met}\, y\, x)^{\eta}) = (\forall x.\, \textsf{ling}\, x \Rightarrow \exists y.\, \textsf{hist}\, y \wedge \textsf{met}\, y\, x)^{\eta}$$

Proof.

$$\begin{aligned}\textsf{\textbf{ev.ling}}\,(\lambda x.\, \textsf{\textbf{a.hist}}^{\triangleright}_{y} \multimap (\textsf{met}\, y\, x)^{\eta}) &= \textsf{\textbf{not}}\,(\textsf{\textbf{a.ling}}_x \multimap \textsf{\textbf{not}}\,(\textsf{\textbf{a.hist}}^{\triangleright}_{y} \multimap (\textsf{met}\, y\, x)^{\eta})) && \textsf{\textbf{ev.ling}}\\ &= \textsf{\textbf{not}}\,(\textsf{\textbf{a.ling}}_x \multimap (\neg\exists y.\, \textsf{hist}\, y \wedge \textsf{met}\, y\, x)^{\eta}) && \textsf{\textbf{not}}\\ &= (\neg\exists x.\, \textsf{ling}\, x \wedge \neg\exists y.\, \textsf{hist}\, y \wedge \textsf{met}\, y\, x)^{\eta} && \textsf{\textbf{not}}\\ &= (\forall x.\, \textsf{ling}\, x \Rightarrow \exists y.\, \textsf{hist}\, y \wedge \textsf{met}\, y\, x)^{\eta} && =\end{aligned}$$

□

In-scope binding for a sentence like *every linguist rubbed her head* can be accomplished along the lines of Fact 3.6. The idea is that we can make a dref-hosting program out of the individuals over which the quantifier quantifies. While I postpone an explicit account of how this happens to Section 3.6.1, the upshot is that the quantifier can bind a pronoun within its scope. Correctly, the result is a pure program returning True iff every linguist rubbed her head.

Fact 3.6 (Every linguist rubbed her head).

$$\textsf{\textbf{ev.ling}}\,(\lambda x.\, x^{\eta\triangleright}_{v} \multimap \textsf{\textbf{pro}}_y \multimap (\textsf{rubbed}\,(\textsf{head}\, y)\, x)^{\eta}) = (\forall x.\, \textsf{ling}\, x \Rightarrow \textsf{rubbed}\,(\textsf{head}\, x)\, x)^{\eta}$$

Proof. Binding simplification, meaning of **ev.ling**. □

Importantly and correctly, the dref introduced by the quantifier is accessible only *within the quantifier's scope*. Much as the dynamically closed quantifier tosses out any drefs contributed by e.g. indefinites within its scope, so the quantifier tosses out its own dref.

3.2.3 Explanatory concerns

When we enrich our meanings to include dynamic quantifiers, we start to rub up on questions of explanatory depth. A potential concern is that there seems to be no principled *truth-conditional* basis (that is, independent of anaphoric phenomena) for analyzing some DPs as nondeterministic indefinites and others as dynamically closed quantifiers. More specifically, any increasing quantifier is *a priori* analyzable as an indefinite or as a dynamically closed quantifier. See (3.4) for an extreme example of the issue: though we've given *a linguist* a nondeterministic Me semantics (on the left), the truth conditions of sentences like *a linguist left* are also consistent with an analysis in terms of a dynamically closed quantifier (on the right). Why choose one and not the other?

(3.4) $\lambda s.\, \{\langle x,\ s\rangle \mid \mathsf{ling}\, x\}$ $\qquad \lambda k s. \{\langle |\mathsf{ling} \cap \{x \mid \exists s'.\, s' \in k\, x\, s\}| \geq 1,\ s\rangle\}$

Two criteria are commonly cited as distinguishing indefinites from "true quantifiers". First, when an indefinite binds a pronoun it doesn't scope over, the pronoun is interpreted *non-maximally*. To wit, (3.5a) is coherent, which implies that the first *it* does not refer to *the* dog owned by Socrates, but rather to *one of* Socrates's dogs. This is not characteristic of "true" quantifiers, e.g. modified numerals (e.g. Kamp & Reyle 1993, van den Berg 1996, Brasoveanu 2007, 2008): example (3.5b) has a strong ring of inconsistency, suggesting that *them* is interpreted maximally when anaphoric to *at least two dogs* (i.e. as roughly synonymous with *the dogs Socrates owns*). Second, indefinites can be interpreted collectively, as in example (3.6a), while "true" quantifiers have a strong tendency to be interpreted distributively, as in example (3.6b) (e.g. Kamp & Reyle 1993, Reinhart 1997, Winter 1998, de Swart 1999, and especially Szabolcsi 1997).

(3.5) a. Socrates has a dog$_i$ and he feeds it$_i$ tasty morsels. Socrates has another dog$_j$ but he only feeds it$_j$ scraps.
(Abbott 2002: 287)

b. Socrates owns at least two dogs$_i$. He feeds them$_i$ tasty morsels. #Socrates owns another dog$_j$, but he only feeds it$_j$ scraps.

(3.6) a. Two linguists drank a beer together.

b. #At least two linguists drank a beer together.

Nevertheless, what qualifies as an indefinite and what doesn't is in part a matter of stipulation.[1] (Related criticisms get leveled at dynamic accounts of presupposition projection, e.g. Heim 1983. See Schlenker 2009 for a recent overview.) In fact, difficulties of this sort characterize *any* theory that attempts to cleave off a semantically distinct class of indefinites—importantly, including static accounts of the *exceptional scope* properties of indefinites (to be explored in Section 4.7).

In Chapter 4 I will argue that the ability of indefinites to take scope out of islands offers another reason, independent of anaphora, to give them a nondeterministic semantics (and, conversely, the inability of "true" quantifiers, including monotone-increasing ones, to scope out of scope islands

1 Syntactic considerations may prove relevant. See Ionin & Matushansky 2006, Hackl 2000, Constant 2012 for discussion.

gives an independent reason to suppose that their semantics is dynamically closed). This provides additional evidence for the nondeterministic Me treatment adopted here.

3.3 Continuations: a denotational perspective on scope-taking

Now that we have irreducibly scopal expressions, in the form of dynamic quantifiers like **ev.ling** and **no.ling**, we need a way to compositionally integrate them into derivations of sentences like *Polly met every linguist*. While the State.Set monad handles nondeterministic and anaphoric side effects, it *doesn't* handle quantifiers in any obvious way. In particular, it makes no provisions for in situ quantification (at least if we stick with our lower-typed lexical entries for transitive verbs), let alone for binding by scopal expressions or inverse scope. The question thus arises (as it does in every compositional framework) how scope-takers interact with their linguistic context.

Continuations provide a well-studied way to compositionally integrate scopal meanings (some key citations are Barker 2002, de Groote 2001, Shan 2002, Shan & Barker 2006, Barker & Shan 2006, 2008, 2014), and turn out to interact with monads in an exceptionally natural way. This section begins with an introduction to continuations and continuized grammars suitable for semanticists with no experience using either. The key idea is that some meanings incur scopal side effects, and that the semantics of combination has to accommodate this. I formalize this using a Continuation monad.

Underlying continuized grammars is a simple, grammar-wide generalization of the sort of scope-taking that semanticists routinely and adeptly use to model quantificational scope and scope ambiguities. The intuitions semanticists have developed for reasoning about scope-taking in LF-based frameworks can be imported wholesale into the continuized setting. To that end, I introduce a variant of Barker & Shan 2008's *tower notation* for constructing and reasoning about derivations in continuized grammars. We'll run through some examples using static quantifiers, i.e. with type $(e \to t) \to t$, and conclude with a discussion of evaluation/Lowering.

3.3.1 Introducing continuations

Let's begin with some simple examples. To keep things as simple as possible, I'll temporarily set aside the State.Set monad in favor of a static semantics, i.e. one in which quantificational DPs have type $(e \to t) \to t$, transitive verbs have type $e \to e \to t$, and so on. This leads to a familiar pickle when a quantificational DP is the object of a transitive verb: functional application isn't defined since the types aren't right: a function with type $(e \to t) \to t$ is not in the domain of a function with type $e \to e \to t$, and vice versa. Moreover, positing a higher type for the transitive verb, e.g. $((e \to t) \to t) \to e \to t$, resolves the type mismatch but does not tell us how a quantified object can scope over a quantified subject, e.g. on the $\forall > \exists$ reading of *a stone lion guards every embassy*.

A number of theories offer solutions to these issues, ranging from Montague's 1974 Quantifying In to May's 1985 LF with Quantifier Raising to Cooper Storage (1983) to Hendriks's 1993 Flexible Types. *Continuations* offer another solution. In a continuized grammar, *every* constituent potentially has a scopal semantics. For example, in a continuized grammar a possible denotation for *saw* is the

"Montague-lift" of the normal $e \to e \to t$ denotation, namely $\textsf{saw}^{\uparrow} = \lambda k.\, k\, \textsf{saw}$. See Definition 3.5.

Definition 3.5 (Lifting).

$$x^{\uparrow} := \lambda k.\, k\, x$$

The argument of a Lifted function is its *continuation*—in other words, its scope. Lifting turns *saw* from a run-of-the-mill two-place predicate into a scope-taker.

This means that in at least some instances, the inputs to semantic combination are *both* scopal, and the semantics of combination isn't functional application, but rather some notion of *scopal* functional application. Of course, simply scoping a *everyone* over *saw* doesn't magically secure a defined result. The DP still receives an argument of type $e \to e \to t$, when it's expecting an argument with type $e \to t$: it's just as if we'd tried to feed saw directly to ⟦every woman⟧:

$$(3.7) \qquad \begin{aligned} &(\lambda k.\, k\, \textsf{saw})\, (\lambda R.\, [\![\text{every woman}]\!]\, (\underbrace{\lambda x.\, R\, x}_{e\to e\to t})) \\ &= [\![\text{every woman}]\!]\, \textsf{saw} \quad ✗ \end{aligned}$$

What went wrong? We failed to be systematic: if *saw* denotes a scopal verb, and *every woman* a scopal DP, shouldn't the result of combining them be a *scopal VP*—itself a function over continuations? That seems reasonable enough: combining two continuized meanings should probably yield a third. See Definition 3.6 for an operation that does this, scoping two meanings over a skeleton assembled with functional application to yield a composite scope-taker. This operation, *scopal functional application* ('SA'), is the mode of combination seen in the continuations-based grammars of Barker 2002, Shan 2002, Shan & Barker 2006, Barker & Shan 2008, 2014.

Definition 3.6 (Scopal functional application).

$$\begin{aligned} \textsf{SA}\; m\, n := \lambda k.\, m\, (\lambda x.\\ n\, (\lambda y.\\ k\, (\textsf{A}\, x\, y))) \end{aligned}$$

Scopal application solves the problem: the continuation k which makes the complex VP scopal can be used to ensure that everything types out. See (3.8), where $\textsf{saw}^{\uparrow}$ and ⟦every woman⟧ are scoped in their surface order to give a composite scope-taker. If k has type $(e \to t) \to t$, the argument to ⟦every woman⟧ has type $e \to t$, and we've managed to secure a defined result:

$$(3.8) \qquad \begin{aligned} &\lambda k.\, (\lambda k.\, k\, \textsf{saw})\, (\lambda R.\, [\![\text{every woman}]\!]\, (\underbrace{\lambda x.\, k\, (R\, x)}_{e\to t})) \\ &= \lambda k.\, [\![\text{every woman}]\!]\, (\lambda x.\, k\, (\textsf{saw}\, x)) \quad ✓ \end{aligned}$$

This solution for how scopal DPs interact with their surroundings—namely, that everything is (or becomes) scopal and combines via SA to yield another scopal expression—turns out to be completely general. For instance, we can use it to combine a scope-taking VP, such as the one we just derived, with a scopal subject as in (3.9) (here k has type $t \to t$) (notice that scoping the scopal subject over the scopal VP, as we do here, gives surface-scope truth conditions; other options are available for inverse scope cases and will be explored in Section 3.6.2):

(3.9) $$\lambda k.\, [\![\text{a man}]\!]\, \underbrace{(\lambda y.\, (\lambda k.\, [\![\text{every woman}]\!]\, (\lambda x.\, k\, (\mathsf{saw}\, x)))\, (\lambda P.\, k\, (P\, y)))}_{e\to t}$$
$$= \lambda k.\, [\![\text{a man}]\!]\, (\lambda y.\, [\![\text{every woman}]\!]\, (\lambda x.\, k\, (\mathsf{saw}\, x\, y)))\quad ✓$$

Likewise, we can use SA to directly combine a ditransitive verb like *gave* with a scopal direct object (here k has type $(e \to e \to t) \to t$):

(3.10) $$\lambda k.\, (\lambda k.\, k\, \mathsf{gave})\, (\lambda D.\, [\![\text{every book}]\!]\, \underbrace{(\lambda x.\, k\, (D\, x))}_{e\to t})$$
$$= \lambda k.\, [\![\text{every book}]\!]\, (\lambda x.\, k\, (\mathsf{gave}\, x))\quad ✓$$

Having built a complex scopal transitive, we're free to keep moving up the tree, scoping meanings as we go until we reach the top.

Though we have not yet addressed inverse scope, the interim conclusion is that a natural way to deal with in situ quantifiers is to suppose that all constituents potentially take scope, and that the mechanism for composing the meanings of syntactically complex constituents scopes the meanings of their parts to yield a composite scopal expression. Formally, this can be summed up in two rules: a Lifting rule for coaxing meanings into scope-takers, along with SA, a mode of combination which scopes two scopal programs to yield a composite scopal expression.

3.3.2 The Continuation monad

The perspective on semantic composition detailed in the previous section—that anything can be scopal and that composition is a fundamentally scopal process—can be given a monadic characterization. This might not be surprising: the Lift operation—a device for making a trivial scope-taker out of a value—looks an awful lot like an injection function, while SA looks an awful lot like a way to extend A along the lines we explored in Chapter 2.

The monad that underwrites scopal composition is the Continuation monad. The Continuation monad deals with *scopal* side effects. Fixing some *result type* ρ, the Continuation monad is as in Definition 3.7 (see e.g. Murthy 1992, Wadler 1994, Shan 2002). I use some slightly different notation for the elements of the monadic triple—i.e. $\langle \mathsf{K}_\rho, \uparrow, \multimap\!\!\bullet \rangle$ in lieu of $\langle \mathsf{M}, \eta, \multimap \rangle$. Aside from the important fact that here we parametrize the type-constructor to result types, the notation's significance is just mnemonic; it visually distinguishes Continuation-monadic programs from programs in other monads. Given a result type ρ, a program in the Continuation monad that returns a value of type α has type $\mathsf{K}_\rho\, \alpha$, a function from functions-from-α-typed-things-to-ρ-typed-things into ρ-typed things. (This may remind you of the type of extensional generalized quantifiers, i.e. $(e \to t) \to t$; indeed, Barker 2002 has emphasized that generalized quantifiers can be thought of as Continuized individuals, type $\mathsf{K}_t\, e$.) Injecting a value a into the Continuation monad means turning a into a trivial scope-taker, i.e. mapping a into the Montague-lift of a, namely $\lambda k.\, k\, a$ (see e.g. Montague 1974, Partee & Rooth 1983, Partee 1986, Barker 2002). Last, sequencing m with $\lambda v.\, \pi$ means returning a composite scopal program by feeding a new continuation to π and scoping m over the result.

Definition 3.7 (The Continuation monad).

$$\begin{aligned} \mathsf{K}_\rho\,\alpha &::= (\alpha \to \rho) \to \rho \\ a^{\uparrow} &:= \lambda k.\, k\, a \\ m_v \multimap \pi &:= \lambda k.\, m\, (\lambda v.\, \pi\, k) \end{aligned}$$

Monadic application in the Continuation monad means *scoping* two scopal programs m and n to yield a composite scopal program whose value is the result of doing functional application on the values returned by m and n. See Fact 3.7. This operation is equivalent to SA, justifying our claim that the Continuation monad underlies continuations-based approaches to composition. To distinguish this operation from monadic application in other monads, I write it '**S**' instead of '**A**'. Like **A** in the State and State.Set monads, **S** is not in general commutative. In terms familiar to linguists: scoping m over n is not in general the same as scoping n over m. Thus, in the Continuation monad, order of evaluation matters (cf. e.g. Plotkin 1975, Shan & Barker 2006).

Fact 3.7 (Application in the Continuation monad).

$$\begin{aligned} \mathbf{S}\, m\, n = \lambda k.\, m\, (\lambda x.&\\ &n\, (\lambda y.\\ &\quad k\, (\mathsf{A}\, x\, y))) \end{aligned}$$

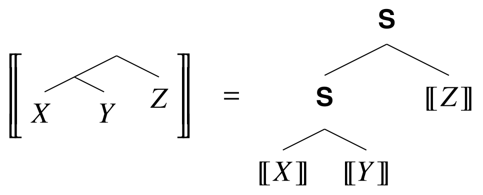
Proof.

$$\begin{aligned} \mathbf{S}\, m\, n &= m_x \multimap n_y \multimap (\mathsf{A}\, x\, y)^{\uparrow} && \mathbf{S} \\ &= m_x \multimap n_y \multimap \lambda k.\, k\, (\mathsf{A}\, x\, y) && \uparrow \\ &= m_x \multimap \lambda k.\, n\, (\lambda y.\, k\, (\mathsf{A}\, x\, y)) && \multimap \\ &= \lambda k.\, m\, (\lambda x.\, n\, (\lambda y.\, k\, (\mathsf{A}\, x\, y))) && \multimap \end{aligned}$$

□

Because the Continuation monad is, after all, a monad, the monad laws and their corollaries still hold (see Appendix B for a proof that the Continuation monad is a monad). In particular, Rebracket allows us to reformulate any complex Continuation-monadic program as a linear sequence of programs. Together with the left-biased rule for monadic application **S**, this entails that syntactic structures are interpreted with a left-to-right bias. An illustration along the lines of (2.1) is given in (3.11). A completely left-branching phrase marker generates an isomorphic analysis tree, and an application of Rebracket reveals that the result is equivalent to a fully linearized analysis, where we run the programs denoted by the terminal nodes in their linear order and do functional application twice on the resulting values. In particular, though X doesn't c-command Z in $\mathcal{T}$, $[\![X]\!]$ is evaluated before $\llbracket Z \rrbracket$ in $\llbracket \mathcal{T} \rrbracket$; simply by virtue of being to Z's left, X scopes over Z.

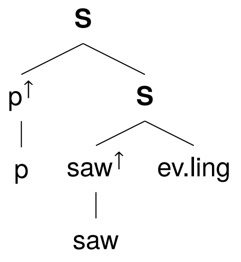
(3.11)



$$= (\textsf{S}\,\llbracket X \rrbracket\,\llbracket Y \rrbracket)_v \multimap \llbracket Z \rrbracket_z \multimap (\mathsf{A}\,v\,z)^{\uparrow} \qquad \textsf{S}$$
$$= \llbracket X \rrbracket_x \multimap \llbracket Y \rrbracket_y \multimap \llbracket Z \rrbracket_z \multimap (\mathsf{A}\,(\mathsf{A}\,x\,y)\,z)^{\uparrow} \qquad \text{Rebracket}$$

A Continuation-monadic derivation for the sentence *Polly saw every linguist* is given in (3.12). For the sake of illustrating how the Continuation monad assembles scopal programs, the derivations up to Section 3.4 will *temporarily* assume that $\rho \coloneqq t$, and that ⟦every linguist⟧ is the static generalized quantifier $\lambda k.\,\forall x.\,\mathsf{ling}\,x \Rightarrow k\,x$, type $\mathsf{K}_t\;e$. As with all the other monadic derivations we've seen, we begin by Lifting the value-denoting expressions into the monad, here via applications of the injection function $\uparrow$. We combine everything up via two applications of **S**, Rebracket into a linear sequence of programs, get rid of the trivial programs via LeftID, and simplify:

(3.12) $\llbracket \text{Polly}^{\uparrow}\ \text{saw}^{\uparrow}\ \text{every linguist} \rrbracket$ =



$$= \mathsf{p}^{\uparrow}_x \multimap \mathsf{saw}^{\uparrow}_f \multimap \mathsf{ev.ling}_y \multimap (f\;y\;\mathsf{p})^{\uparrow} \qquad \text{Rebracket}$$
$$= \mathsf{ev.ling}_y \multimap (\mathsf{saw}\;y\;\mathsf{p})^{\uparrow} \qquad \text{LeftID}$$
$$= \mathsf{ev.ling}_y \multimap \lambda k.\,k\,(\mathsf{saw}\;y\;\mathsf{p}) \qquad \uparrow$$
$$= \lambda k.\,\mathsf{ev.ling}\,(\lambda y.\,k\,(\mathsf{saw}\;y\;\mathsf{p})) \qquad \multimap$$
$$= \lambda k.\,\forall y.\,\mathsf{ling}\;y \Rightarrow k\,(\mathsf{saw}\;y\;\mathsf{p}) \qquad \mathsf{ev.ling}$$

The result is a $\mathsf{K}_t\;t$ program which gives ev.ling scope over the sentence. Despite the apparent type mismatch between transitive verb and scopal object, composition is successful.

3.3.3 The tower notation

As semanticists, we're pretty comfortable with scope-taking, and we're used to the idea that when we scope two expressions in a given order, the scopally relevant parts of their meanings end up interleaved in the order in which they were scoped, while the scope-less parts of their meanings filter down to build the predicate-argument skeleton over which the scopal parts scope. Scope in the Continuation monad works in an analogous way, the only difference being that the result of combining two expressions is *itself* scopal, i.e. a function over continuations.

We can draw on this intuition to reason about what happens in continuized derivations, even in relatively complicated cases like *a teacher read two stories to every child*. Since the scope-takers stay at the top, interleaved in their surface order (by stipulation), and the scopally inert expressions filter down to build the propositional skeleton over which the scope-takers scope, the Continuation-monadic derivation of the sentence as a whole must amount to (3.13).

(3.13) $\lambda k.\, [\![\text{a teacher}]\!]\, (\lambda x.\, [\![\text{two stories}]\!]\, (\lambda y.\, [\![\text{every child}]\!]\, (\lambda z.\, k\, (\mathsf{read}\; y\; (\mathsf{to}\; z)\; x))))$

What I mean to suggest is that doing derivations in continuized grammars is easier and more familiar than you might have supposed. In particular, there's a handy short-cut which capitalizes on our intuitions about scope-taking to allow us to quickly and accurately construct and reason about continuized derivations. The short-cut for doing continuized derivations, introduced in Barker & Shan 2008, 2014 and elaborated here, is the *tower notation*.

I'll make use of two kinds of towers: towers for types, see Definition 3.8, and towers for meanings, see Definition 3.9. In both cases, the general idea is to represent a program in the Continuation monad in a way that distinguishes its scopal side effects from the value it returns. For instance, the type in Definition 3.8 can be read as "returns a value of type α in an evaluation context with type ρ", i.e. as distinguishing the type of the program's value from the type of the larger computation over which it takes scope.[2] Second, the meaning tower in Definition 3.9 visually distinguishes the value returned by a Continuation-monadic program, which lives on the bottom level of the tower, from any side effects that evaluating the program incurs, which live on the upper level of the tower. (Think of f as standing in for an arbitrary, possibly empty string of symbols.)

Definition 3.8 (Tower types).

$$\frac{\rho}{\alpha} ::= \mathsf{K}_\rho\, \alpha ::= (\alpha \to \rho) \to \rho$$

Definition 3.9 (Tower meanings).

$$\frac{f\,[\,]}{x} \;:=\; \lambda k.\, f\,[k\,x]$$

Before I get to some examples of the tower notation in action, a few words on the notational conventions I'll use to represent relevant linguistic information and construct derivations. Following Barker & Shan 2008, 2014, I associate each expression with a stacked triple. From top to bottom, a stacked triple specifies an expression's type, phonological/orthographic form, and meaning. For instance, functional application of some *left-exp* that denotes x and some *right-exp* that denotes y

2 Unlike Barker & Shan 2008, 2014, I use towers to represent *types*, but not syntactic categories; the close correspondence between categories and types in categorial grammars makes this shift in perspective immediate. In addition, I'm using bipartite rather than tripartite towers. In fact, the Continuation monad properly only allows for bipartite towers (cf. Wadler 1994: 48), though a slight and harmless generalization allows for tripartite towers. More on this in Section 3.6.3.

gives an expression pronounced '*left-exp right-exp*' and meaning $\mathsf{A}\,x\,y$, as in Fact 3.8. An example derivation using two instances of functional application to derive a meaning for *Sam saw Elliott* is given in (3.14).

Fact 3.8 (Functional application). $\{\gamma,\ \delta\} \coloneqq \{\alpha,\ \alpha \to \beta\}$, for some types α and β:

$$\left(\begin{array}{cc} \gamma & \delta \\[1em] \textit{left-exp} & \textit{right-exp} \\[1em] x & y \end{array}\right) = \begin{array}{c} \beta \\[1em] \textit{left-exp right-exp} \\[1em] \mathsf{A}\,x\,y \end{array}$$

(3.14)
$$\left(\begin{array}{c} e \\[1em] \text{Sam} \\[1em] \mathsf{s} \end{array} \left(\begin{array}{cc} e \to e \to t & e \\[1em] \text{saw} & \text{Elliott} \\[1em] \mathsf{saw} & \mathsf{e} \end{array}\right)\right) = \begin{array}{c} t \\[1em] \text{Sam saw Elliott} \\[1em] \mathsf{saw\,e\,s} \end{array}$$

Now for some scopal examples. The (static) *every linguist*, whose linear type can be written $\mathsf{K}_t\, e$ or $(e \to t) \to t$, corresponds to either of the stacked triples in (3.15). Notice for the triple on the left that we convert $\mathsf{ev.ling}$ into a tower via the following routine λ-theoretic equivalences: $\mathsf{ev.ling} = \lambda k.\, \mathsf{ev.ling}\, k = \lambda k.\, \mathsf{ev.ling}\,(\lambda x.\, k\, x)$. The type tower says that *every linguist* contributes an individual, type e, to a larger program of type t, which seems sensible enough. The meaning tower distinguishes the scopal side effects that inhere in $\mathsf{ev.ling}$—namely, the quantification over linguists x—from the value it contributes to the larger computation—namely, x (for each quantified-over linguist x).

(3.15)
$$\begin{array}{c} \dfrac{t}{e} \\[1em] \text{every linguist} \\[1em] \dfrac{\mathsf{ev.ling}\,(\lambda x.\,[\,])}{x} \end{array} = \begin{array}{c} \dfrac{t}{e} \\[1em] \text{every linguist} \\[1em] \dfrac{\forall x.\, \mathsf{ling}\,x \Rightarrow [\,]}{x} \end{array}$$

As another example, Fact 3.9 shows how to express the effect of the Continuation monad's injection function $\uparrow$ in the tower notation. To the left of the equals sign is an *exp* which denotes a value a with type α. Applying $\uparrow$ gives an *exp* whose meaning is the tower notation's equivalent of $\lambda k.\, k\, a$, and whose type is the tower notation's equivalent of $\mathsf{K}_\rho \alpha$ (fixing some result type ρ). Notice in particular how the fact that $a^\uparrow$ is a pure program in the Continuation monad corresponds to the upper level of the resulting tower being devoid of scopal side effects: $a^\uparrow$ contributes nothing to a computation besides the simple value a.

Fact 3.9 (Lifting in the tower notation). Fixing some result type ρ:

$$\begin{pmatrix} \alpha \\ \\ exp \\ \\ a \end{pmatrix}^{\uparrow} = \begin{array}{c} \dfrac{\rho}{\alpha} \\ \\ exp \\ \\ \dfrac{[\,]}{a} \end{array}$$

The great payoff of the tower notation is the ease with which it allows us to compositionally assemble complex scopal programs. The towers version of scopal functional application is given in Fact 3.10. On the bottom level, meanings combine via functional application, and on the top level, the scopal effects compose with a linear bias. There is no need to Rebracket, apply LeftID, and so forth; these steps are built into the workings of the tower notation.[3]

Fact 3.10 (Tower combination). $\{\gamma,\ \delta\} \coloneqq \{\alpha,\ \alpha \to \beta\}$, for some types α and β:

$$\begin{pmatrix} \dfrac{\rho}{\gamma} & \dfrac{\rho}{\delta} \\ \\ \textit{left-exp} & \textit{right-exp} \\ \\ \dfrac{f\,[\,]}{x} & \dfrac{g\,[\,]}{y} \end{pmatrix} = \begin{array}{c} \dfrac{\rho}{\beta} \\ \\ \textit{left-exp right-exp} \\ \\ \dfrac{f\,[g\,[\,]]}{\mathsf{A}\,x\,y} \end{array}$$

As for types, the bottom level works as in Definition 3.8: we require one of the values to be in the domain of the other. On the top level, we require *matching types*. In other words, scopal functional application presupposes that the programs being combined presuppose the same result type. It will be impossible, for instance, for one tower to anticipate that it's participating in the derivation of a sentence (type t), while the other tower thinks it's participating in the derivation of a DP (type e) (NB: the tower notation *reflects* rather than stipulates this fact about composing meanings).

3.3.4 Practice derivations

Let's practice a bit with the tower notation using static generalized quantifiers before we fold in the State.Set monad. We begin with a towers-based derivation for *Polly saw every linguist*, the case we already derived more clunkily in (3.12). As in the monadic derivation, we first Lift the value-denoting *Polly* and *saw* into the Continuation monad via applications of $\uparrow$. We combine everything up via two

3 For readability's sake, I'm allowing for a bit of ambiguity in the way combination is notated (cf. Fact 3.8). In general, it'll either be explicitly stated or obvious from the context how I intend a combination to be interpreted.

instances of scopal functional application to yield a composite sentential tower, type $\mathsf{K}_t\ t$.

(3.16)
$$\left(\begin{array}{c} \frac{t}{e} \\[1em] \text{Polly} \\[1em] \frac{[\,]}{\mathsf{p}} \end{array} \left(\begin{array}{cc} \frac{t}{e \to e \to t} & \frac{t}{e} \\[1em] \text{saw} & \text{every linguist} \\[1em] \frac{[\,]}{\mathsf{saw}} & \frac{\mathsf{ev.ling}\ (\lambda x.\,[\,])}{x} \end{array}\right)\right) = \begin{array}{c} \frac{t}{t} \\[1em] \text{Polly saw every linguist} \\[1em] \frac{\mathsf{ev.ling}\ (\lambda x.\,[\,])}{\mathsf{saw}\ x\ \mathsf{p}} \end{array}$$

The derived tower, on the right, is equivalent to the linearized term derived in (3.12). But this derivation is a good deal simpler. There is no need to construct a monadic parse, Rebracket, apply LeftID, and iteratively simplify. On the bottom level, composition is functional application of values. On the top level, effects are composed with a leftward bias (though in this case the only expression associated with any scopal effects is *every linguist*). The clean separation between the scopal parts of a Continuation-monadic program and the value it returns makes the derivation quick and straightforward.

Derivations with more than one quantifier go about as smoothly. A schematic case with static generalized quantifiers in both subject and object position is derived in (3.17). The derivation is basically analogous to (3.16), though it involves one less application of Lift (since the subject is already scopal). As before, the values on the bottom levels combine via functional application, and the side effects on the scopal tier compose in their linear order. The result is a tower, type $\mathsf{K}_t\ t$, which gives the subject quantifier $\mathrm{Q}_{\mathrm{subj}}$ scope over the object quantifier $\mathrm{Q}_{\mathrm{obj}}$.

(3.17)
$$\left(\begin{array}{c} \frac{t}{e} \\[1em] \mathrm{Q}_{\mathrm{subj}} \\[1em] \frac{\mathsf{Q}_{\mathsf{subj}}\ (\lambda x.\,[\,])}{x} \end{array} \left(\begin{array}{cc} \frac{t}{e \to e \to t} & \frac{t}{e} \\[1em] \text{saw} & \mathrm{Q}_{\mathrm{obj}} \\[1em] \frac{[\,]}{\mathsf{saw}} & \frac{\mathsf{Q}_{\mathsf{obj}}\ (\lambda y.\,[\,])}{y} \end{array}\right)\right) = \begin{array}{c} \frac{t}{t} \\[1em] \mathrm{Q}_{\mathrm{subj}}\ \text{saw}\ \mathrm{Q}_{\mathrm{obj}} \\[1em] \frac{\mathsf{Q}_{\mathsf{subj}}\ (\lambda x.\,\mathsf{Q}_{\mathsf{obj}}\ (\lambda y.\,[\,]))}{\mathsf{saw}\ y\ x} \end{array}$$

Clearly, we'll be able to give analogous derivations for sentences with an arbitrary number of scopal expressions (e.g. *a teacher read two stories to every child*). The result will inevitably be a two-level tower, whose top level has the scopal parts interleaved in their linear order, and whose bottom level has the predicate-argument skeleton over which the scopal bits scope.

3.3.5 Evaluation

A couple questions arise at this point. For one, it isn't at all clear how to derive *inverse*-scope readings; the stipulation that evaluation happens left-to-right yields surface-scope readings in every case we've considered so far. In fact, inverse scope derivations are already present in the grammar; we just need to look a bit harder. We'll see how this works in Section 3.6.2.

Another issue is pressing. If composition gives every scope-taker scope over *everything* to its right, it would seem to follow that scope-takers should have unbounded rightward scope. This can't be correct. For example, imagine that a negative quantifier like *no linguist* enters a continuized derivation at some point. If *no linguist* could have unbounded rightward scope, we'd incorrectly, catastrophically end up placing everything we'd yet to say inside a downward-entailing environment! See for instance (3.18), a possible analysis (given what we've said so far) for *no linguist came to the party; it was raining*. The negation inhering in no.ling $:= \lambda k.\, \neg\exists x.\, \mathsf{ling}\, x \wedge k\, x$ incorrectly ends up scoping over rained!

(3.18)
$$\begin{array}{c} \dfrac{t}{t} \\[2ex] \text{no linguist came to the party; it was raining} \\[2ex] \dfrac{\mathsf{no.ling}\,(\lambda x.\,[\,])}{\mathsf{came}\ x \wedge \mathsf{rained}} \end{array}$$

Evidently, we need a way to close off derivations, a way to prevent scopal expressions' scopes from indefinitely expanding rightward. In another manner of speaking, we need a way to *evaluate* towers and we need to ensure, at the very least, that evaluation is *obligatory* at sentence boundaries. (I'd emphasize that this is not something unique to using continuations for scope. The corresponding stipulation in LF-based grammars is that you cannot QR out of a tensed clause, and certainly not to a position of text-level scope.) A rule for evaluation (taken from Shan & Barker 2006, Barker & Shan 2008; see especially the latter for pertinent discussion about delimiting rightward scope-taking), is given in Definition 3.10, and the towers version in Fact 3.11. Evaluation means closing off a continuized expression's scope by feeding it a *trivial* continuation k_0—here, the identity function.[4]

Definition 3.10 (Lower/evaluation).

$$m^{\downarrow} := m\,(\lambda a.\, a)$$

Fact 3.11 (Lowering/evaluating towers).

$$\left(\begin{array}{c} \dfrac{\alpha}{\alpha} \\[2ex] exp \\[2ex] \dfrac{f\,[\,]}{a} \end{array}\right)^{\downarrow} = \begin{array}{c} \alpha \\[2ex] exp \\[2ex] f\,[a] \end{array}$$

Here is an example of how Lowering works. In (3.19) I apply ↓ to the tower we derived for *Polly saw every linguist*. Evaluating this tower means collapsing it into a single level by applying it to the

4 See Groenendijk & Stokhof 1990, de Groote 2006, Szabolcsi 2003 for related notions of evaluation via application to a trivial continuation. Evaluation also corresponds to "emptying the store" in Cooper Storage-based approaches to scope-taking (Cooper 1983).

identity function. This correctly delivers $\textsf{True}$ iff Polly saw every linguist. Similarly, Lowering the two-quantifier tower derived in (3.17) gives $\textsf{Q}_{\textsf{subj}}\,(\lambda x.\,\textsf{Q}_{\textsf{obj}}\,(\lambda y.\,\textsf{saw}\,y\,x))$. The quantificational force in each quantifier is discharged, and the derivation is complete. In both cases, the forward march of rightward scope-taking is halted by the application of Lower.

(3.19)
$$\left(\begin{array}{c} \dfrac{t}{t} \\[1ex] \text{Polly saw every linguist} \\[1ex] \dfrac{\textsf{ev.ling}\,(\lambda x.\,[\,])}{\textsf{saw}\,x\,\textsf{p}} \end{array}\right)^{\downarrow} = \begin{array}{c} t \\[1ex] \text{Polly saw every linguist} \\[1ex] \textsf{ev.ling}\,(\lambda x.\,\textsf{saw}\,x\,\textsf{p}) \end{array}$$

Notice that re-Lifting the derived type-t expressions will leave the quantifiers on the ground story, forever subsumed into a boolean value. Thus, if evaluation of sentences is obligatory, we correctly predict, for instance, that *no linguist came to the party, and it was raining* isn't true in a world where linguists came to the party, but it wasn't raining (as $\neg\exists x.\,\textsf{ling}\,x \wedge \textsf{came}\,x \wedge \textsf{rained}$ would have it): if *no linguist came to the party* is obligatorily evaluated, the scopal effects of $\textsf{no.ling}$ are forever discharged at that point and unable to acquire scope over subsequent sentences.

3.4 Monadic composition via continuations

We've been using continuations and **S** to assemble complex scopal programs. Besides recognizing that the scopal compositional regime can be formulated as a monadic grammar, we have not as yet related the continuations-based perspective on scope-taking to the work we did in Chapter 2. Concretely, we have not seen how we might combine continuations with the regime for side effects we converged on there—namely, the State.Set monad.

This section fills that gap. I pursue a novel connection between using continuations for scopal composition in the presence of what I call an "underlying monad"—for example the State.Set monad (novel, that is, within the linguistics literature; the work here is closely related to Wadler's 1994 research on the relationship between continuations and monads and to Liang et al.'s 1995 work on monad transformers). The upshot is a generalization of the continuations-based semantics we've been working with so far, in which scopal composition interfaces seamlessly with an arbitrary underlying monad, giving a way to assemble programs with side effects in the presence of irreducibly scopal expressions.

3.4.1 Interlude: the ContT monad transformer

Previously, we combined side effects regimes (i.e. monads) by identifying a monad transformer for one and applying it to the other. Something similar can be done to combine scope-taking with side effects. That is, there is a ContT monad transformer which we can apply to an arbitrary *underlying* monad to get an enriched regime for handling scopal side effects in addition to the side effects handled by the underlying monad. Here is the ContT monad transformer (Liang et al. 1995; see also Wadler 1994 for a related discussion about integrating monads and continuations).

Definition 3.11 (ContT). Given some $\langle \mathsf{M}_1, \eta_1, \multimap_1 \rangle$:

$$\begin{aligned} \mathsf{K}_{2\rho}\alpha &::= (\alpha \to \mathsf{M}_1\rho) \to \mathsf{M}_1\rho \\ a^{\uparrow_2} &:= (\multimap_1)\, a^{\eta_1} = \lambda k.\, k\, a \\ m_v \multimap\!\!\bullet_2\, \pi &:= \lambda k.\, m\, (\lambda v.\, \pi\, k) \end{aligned}$$

For any monad $\mathcal{M}$, ContT applied to $\mathcal{M}$ turns out to be... *a Continuation monad*. The only difference to be found between the basic Continuation monad and ContT $\mathcal{M}$ (for any monad $\mathcal{M}$) is in the latter's more specific characterization of its type constructor $\mathsf{K}_{2\rho}\alpha$: ContT $\mathcal{M}$ is a monad for characterizing scopal programs which return programs in the underlying monad $\mathcal{M}$.

Though we will not make *direct* use of ContT in what follows (though cf. Appendix A), it points the way to the synthesized perspective we're after. Notice in particular that Lift in the transformed monad (i.e. $\uparrow_2$) is defined in terms of (the composition of) the injection and sequencing operations from the underlying monad (i.e. η_1 and $\multimap_1$): the function of η_1 is to turn a value into a program m in the underlying monad, and the function of $\multimap_1$ is to turn m into a scope-taker M in the transformed monad. As indicated in the definition above, the result is equivalent (modulo types) to the Lift operation in the Continuation monad (and in earlier continuations-based approach to natural language scope-taking):

$$\begin{aligned} (\multimap_1)\, a^{\eta_1} &= \lambda k.\, a_v^{\eta_1} \multimap_1 k\, v \\ &= \lambda k.\, k\, a \qquad\qquad \text{LeftID} \end{aligned}$$

In other words, given an underlying monad $\mathcal{M}$, the Lift operation we've been working with can be *decomposed* into the fundamental operations (injection and sequencing) that characterize $\mathcal{M}$.

3.4.2 Monadic Lift

I suggest that this is the key to unlocking the connection between side effects in an underlying monad and scope-taking. Consider that, for any monad $\mathcal{M}$, the type of $\mathcal{M}$'s sequencing operator is $\mathsf{M}\alpha \to (\alpha \to \mathsf{M}\beta) \to \mathsf{M}\beta$. Now consider the type of the sequencing operator applied to its first argument: $(\alpha \to \mathsf{M}\beta) \to \mathsf{M}\beta$. This looks a lot like the type of a continuized meaning; in fact it is fully equivalent to the type $\mathsf{K}_{\mathsf{M}\beta}\alpha$ (NB: we've shifted back to the Continuation monad way of talking from the ContT way of talking). In other words, we can identify the second argument of the sequencing operator *as a continuation/scope*.

More generally, a given monad's sequencing operator can be used to turn *any* program in that monad into a scope-taker. That is, we can (and will) treat the sequencing operator as a monadic Lift operation. See Definition 3.12 and Fact 3.12 for the towers version.[5]

5 As I noted in fn. 10, every monad transformer implies a Lifting operation, a way to turn programs in the to-be-transformed monad $\mathcal{M}_1$ into programs in the transformed monad $\mathcal{M}_2$. The Lift operation for ContT applied to an arbitrary monad $\mathcal{M}$, written '$\uparrow$', is precisely $\mathcal{M}$'s sequencing operation $\multimap$.

Definition 3.12 (Monadic Lift). Given some $\langle \mathsf{M}, \eta, \multimap \rangle$:

$$m^{\uparrow} := (\multimap)\, m$$

Fact 3.12 (Monadic Lift in the tower notation). Given some $\langle \mathsf{M}, \eta, \multimap \rangle$ and some β:

$$\begin{pmatrix} \mathsf{M}\alpha \\ \\ exp \\ \\ m \end{pmatrix}^{\uparrow} = \begin{array}{c} \dfrac{\mathsf{M}\beta}{\alpha} \\ \\ exp \\ \\ \dfrac{m_{\nu} \multimap [\;]}{\nu} \end{array}$$

Now we can observe that the version of Lift given previously (i.e. where the Lift of a is $\lambda k.\, k\, a$) is actually a special case of this more general perspective. Specifically, that version of Lift is what you get when you fix the underlying monad to the *Identity* monad. Sequencing in the Identity monad is simple application of a function to a value. In other words, in the Identity monad, $(\multimap) := \lambda mk.\, k\, m$. Thus, applying sequencing to some Identity-monadic m gives $\lambda k.\, k\, m$.

Thus, the present perspective *generalizes* extant continuations-based analyses of scope-taking. Whereas previous analyses work implicitly with an underlying Identity monad, we will allow for other underlying monads (in particular, State.Set).

3.4.3 Monadic Lowering

The last thing required to complete the shift is a rule for evaluation/Lowering. That is, we need to identify a trivial continuation/scope that we can use to finish off derivations. Let's reason backward. I have argued that the Identity monad underlies the previous continuations-based semantics. There, the trivial continuation k_0 implicated in Lower was the identity function—i.e. the Identity monad's η. This is more than suggestive: given any underlying monad $\mathcal{M}$, the trivial continuation used for monadic Lowering will be $\mathcal{M}$'s η operator. See Definition 3.13 and Fact 3.13 for the towers version.

Definition 3.13 (Monadic Lowering). Given some $\langle \mathsf{M}, \eta, \multimap \rangle$:

$$m^{\downarrow} := m\, \eta$$

Fact 3.13 (Monadic Lowering in towers).

$$\begin{pmatrix} \dfrac{\mathsf{M}\alpha}{\alpha} \\ \\ exp \\ \\ \dfrac{f\,[\;]}{a} \end{pmatrix}^{\downarrow} = \begin{array}{c} \mathsf{M}\alpha \\ \\ exp \\ \\ f\,[a^{\eta}] \end{array}$$

Monadic Lower and Lift can be composed to give what's known in the computer science literature on delimited continuations as a *Reset* (e.g. Danvy & Filinski 1990). A Reset discharges a continuized program's scopal effects via Lower and then re-shifts the resulting monadic program into a new scopal program via Lift.

Definition 3.14 (Reset).

$$m^{\downarrow\uparrow}$$

In Chapters 4 and 5 I will argue that Resets explain why side effects (such as nondeterminism and state) can take scope out of scope islands.

3.4.4 Monadic application via scope

For any monad $\mathcal{M}$ determining a monadic Lift (in terms of $\mathcal{M}$'s sequencing operation) and a monadic Lower (in terms of $\mathcal{M}$'s injection function), the scopal application rule **S** properly subsumes $\mathcal{M}$'s monadic application operation **A**, in the specific sense that **A** is derivable from **S**, Lift, Lower:

Fact 3.14 (**S** subsumes **A**). Given some $\langle \mathsf{M}, \eta, \multimap \rangle$ determining some **A**:

$$(\mathbf{S}\, m^{\uparrow} n^{\uparrow})^{\downarrow} = \mathbf{A}\, m\, n$$

Proof.

$$\begin{aligned}
\left(\frac{m_x \multimap [\,]}{x}\quad\frac{n_y \multimap [\,]}{y}\right)^{\downarrow} &= \left(\frac{m_x \multimap n_y \multimap [\,]}{\mathsf{A}\, x\, y}\right)^{\downarrow} && \mathbf{S}\\
&= m_x \multimap n_y \multimap (\mathsf{A}\, x\, y)^{\eta} && \downarrow\\
&= \mathbf{A}\, m\, n && \mathbf{A}
\end{aligned}$$

$\square$

An important corollary of this result is that any number of applications of **A** can be simulated just as well via the same number of applications of **S**. This is easy to see inductively: simply use Lift and **S** to combine any number of $\mathcal{M}$ programs, and then Lower. Functional application happens on the bottom levels, sequencing happens on the top level (with the same linear bias imposed by the monadic grammars we looked at in Chapter 2), and Lowering yields a trivial program over which the sequenced bits are sequenced. The result will be equivalent to a program assembled exclusively out of **A** applications.

In addition to being able to simulate any monadic application function **A**, however, **S** allows us to compositionally integrate irreducibly scopal programs such as **ev.ling**. Generalizing to the worst case, as it were, we only need scopal combination. Continuized combination plus monadic Lift and Lower can do the work of **A**, but also allows for inherently scopal things like quantifiers to compositionally participate in derivations. Thus, aside from vanilla functional application, our *sole* mode of combination will be **S**.

3.4.5 Summing up

The following definition and table summarizes the remarks in this section, as well as the relationship of the proposal on offer here to previous analyses in terms of continuations:

Definition 3.15. A *grammar* is a pair $\langle \textsf{S}, \mathcal{M} \rangle$ of scopal application ($\textsf{S}$) and an underlying monad $\langle \mathsf{M}, \eta, \multimap \rangle$ such that $\uparrow := (\multimap)$ and $k_0 := \eta$.

Theory	$m^{\uparrow}$	k_0
Previous work	$\lambda k.\, k\, m$	$\lambda a.\, a$
My proposal	$\lambda k.\, m \multimap k$	η

Monads and continuations have both been discussed in the context of natural language semantics before. What has not been been emphasized is that the continuations-based apparatus for scopal composition can be seamlessly integrated with an underlying monad in the service of building scopal programs which return programs in the underlying monad. The essential insight for forging this connection is an acknowledgement that the second argument of the underlying monad's sequencing operation $\multimap$ can be identified with a continuation, which builds a bridge between sequencing in the underlying monad and scopal composition using continuations.

The upshot is that, given some underlying monad $\langle \mathsf{M}, \eta, \multimap \rangle$ (and possibly *multiple* underlying monads; see Section 5.4), a complete grammar capable of expressing and composing *everything* I will express and compose in what follows can be given in terms of three simple combinators: $\textsf{S}$, η, and $\multimap$ (and combinators derivatively built from these primitives; see Steedman 2000 for a thorough overview of combinatory categorial grammar and Shan & Barker 2006 for a system more closely related than Steedman's to the one I'll be using here).

3.5 Monadic dynamics with scope

As we observed in the previous section, extant continuations-based semantics for natural language have implicitly presupposed an underlying Identity monad. We will jettison this presumption and allow for underlying monads distinct from the Identity monad. In particular, to handle state-changing and nondeterminism, we will suppose that the underlying monad is the State.Set monad. This section details how this goes, providing concrete examples of how the grammar motivated in Chapter 2 gets folded into the new scopal compositional regime.

3.5.1 Integrating the State.Set monad

I spent Chapter 2 motivating the idea that the State monad is a useful tool for modeling anaphora in natural language, the Set monad is a useful tool for modeling nondeterminism, and the State.Set monad is a useful tool for modeling anaphora and nondeterminism in one fell swoop. This section folds the State.Set semantics into the Continuation monad along the lines sketched in the previous section. The payoff is an extension of the State.Set semantics, into a general compositional scheme that integrates scope-taking with anaphoric and nondeterministic side effects. As a reminder, the State.Set monad was defined as follows:

$$\begin{aligned}
\mathsf{M}\alpha &::= s \to (\alpha \times s) \to t \\
a^{\eta} &:= \lambda s.\, \{\langle a,\ s\rangle\} \\
m_v \multimap \pi &:= \lambda s. \bigcup_{\langle a, s'\rangle \in m\,s} \pi[a/v]\, s'
\end{aligned}$$

Thus, our Lift operation $\uparrow$ has the semantics of the State.Set monad's sequencing operation, and the semantics of the Lowering operation $\downarrow$ means applying a tower to the State.Set monad's injection function η.

3.5.2 The three grades of monadic involvement

We have a space of values, a space of State.Set programs, a space of meanings that take scope (over State.Set programs). And we have a way to upgrade values into programs and programs into scope-takers. This section offers some case studies of how this goes, i.e. of the relationships between values, programs in the underlying monad, and scope-takers.

We begin with value-denoting things like *Polly*, *saw*, *mom*, *and*, and so on—things that we have no reason to suppose have either nondeterministic, anaphoric, or scopal side effects. An example for *Polly* is given below. We inject *Polly* into the State.Set monad with η, then sequence the result with some arbitrary continuation using $\uparrow$. Try as we might, we can't get anything onto the continuized tier: the final equivalence (the result of an application of LeftID) reveals that the result is equivalent to a Montague-lifted value. The same goes, mutatis mutandis, for any value-denoting thing: applying η and then $\uparrow$ to any value a yields an expression equivalent to $\lambda k.\, k\, a$. Notice that the result type, though polymorphic, is restricted to State.Set-monadic programs $\mathsf{M}\alpha$: because the semantics of $\uparrow$ is the semantics of $\multimap$, Lift presupposes that the continuation will be a function into a State.Set-monadic program.

(3.20)
$$\left(\begin{array}{c} e \\[1ex] \text{Polly} \\[1ex] \mathsf{p} \end{array}\right)^{\eta\uparrow} = \left(\begin{array}{c} \mathsf{M}e \\[1ex] \text{Polly} \\[1ex] \mathsf{p}^{\eta} \end{array}\right)^{\uparrow} = \begin{array}{c} \dfrac{\mathsf{M}\alpha}{e} \\[1ex] \text{Polly} \\[1ex] \dfrac{\mathsf{p}^{\eta}_{x} \multimap [\,]}{x} \end{array} = \begin{array}{c} \dfrac{\mathsf{M}\alpha}{e} \\[1ex] \text{Polly} \\[1ex] \dfrac{[\,]}{\mathsf{p}} \end{array}$$

Next, we look at cases where we have reason to suppose that a meaning has nondeterministic or anaphoric side effects, but no irreducibly scopal side effects. In contrast with value-denoters such as *Polly*, these meanings are irreducibly monadic and cannot be derived from values. An example for the State.Set program denoted by *a linguist*, namely **a.ling** ($= \lambda s.\, \{\langle x,\ s\rangle \mid \mathsf{ling}\, x\}$), is given below. As in the previous case, we end up with a tower whose value is type e and whose result type $\mathsf{M}\alpha$ (for some α). This time, however, because **a.man** has side effects in the State.Set monad (specifically, nondeterministic side effects), there is no possibility of reconstructing it into a simple value that lives on the tower's bottom level.

(3.21)
$$\left(\begin{array}{c} \mathsf{M}e \\[1ex] \text{a linguist} \\[1ex] \textbf{a.ling} \end{array}\right)^{\uparrow} = \begin{array}{c} \dfrac{\mathsf{M}\alpha}{e} \\[1ex] \text{a linguist} \\[1ex] \dfrac{\textbf{a.ling}_x \multimap [\,]}{x} \end{array}$$

Last, we look at irreducibly scopal meanings such as **ev.ling** and **no.ling**. These meanings must be defined directly as scopal—i.e. continuized—objects. They cannot be derived from State.Set programs, let alone from values. An example for *every linguist* is given below. As in the previous two cases, the tower's value is type *e*, and its result type is a program in the State.Set monad.

(3.22)
$$\begin{array}{c} \dfrac{\mathsf{M}t}{e} \\[1ex] \text{every linguist} \\[1ex] \dfrac{\textbf{ev.ling}\ (\lambda x.\,[\,])}{x} \end{array}$$

Notice that while universal DPs have a fixed type, Lifted values and indefinites are *polymorphic*, presupposing only that their result type is some sort of program in the State.Set monad. This polymorphism follows immediately from the polymorphism of $\multimap$; e.g. there is nothing inherent to the meaning of *a linguist* or $\multimap$ which forces us to be any more specific (though, the *e* on the bottom level of the tower-type guarantees that *a linguist*$^{\uparrow}$ is a scopal individual, the same as *John* or *every linguist*). By contrast, the meaning of *every linguist* tells us that it *must* return a $\mathsf{M}t$: **ev.ling** is only defined at some k if k is a function into dynamic propositions, and the result of applying $[\![$every linguist$]\!]$ to such a k inevitably returns another dynamic proposition. The polymorphism of Lifted indefinites turns out to play an important role later, specifically in Sections 4.5.2, 4.5.3.

A summary of lexical entries is given in Table 3.1. Two brief comments. First, these entries make no provisions for binding; we'll see how discourse referents find their way onto the stack in Section 3.6.1. Finally, will see in Section 3.6.3 how meanings for complex DPs are derived from lexical entries for the determiners along with an apparatus for assembling relative clauses and nominal restrictors.

3.5.3 Composition

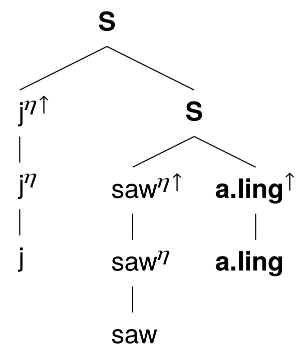
We're now ready to give a derivation for *John saw a linguist*. This derivation, in (3.23), proceeds analogously to (3.16). We apply Lift to the subject, verb, and object. Notice that for the subject and

Item	**Meaning**	**Type**
John	j	e
left	left	$e \to t$
mom	mom	$e \to e$
saw	saw	$e \to e \to t$
and	and	$t \to t \to t$
a linguist	$\textbf{a.ling} := \lambda s.\, \{\langle x,\, s\rangle \mid \mathsf{ling}\, x\}$	$\mathsf{M}e$
she	$\textbf{pro} := \lambda s.\, \{\langle s_\top,\, s\rangle\}$	$\mathsf{M}e$
not	$\textbf{not} := \lambda ms.\, \{\langle \neg\exists s'.\, \langle \mathsf{True},\, s'\rangle \in m\, s,\, s\rangle\}$	$\mathsf{M}t \to \mathsf{M}t$
if	$\textbf{if} := \lambda mn.\, \textbf{not}\, (m_p \multimap (\textbf{not}\, n)_q \multimap (p \wedge q)^\eta)$	$\mathsf{M}t \to \mathsf{M}t \to \mathsf{M}t$
every linguist	$\textbf{ev.ling} := \dfrac{\textbf{not}\,(\textbf{a.ling}_x \multimap \textbf{not}\,[\ ])}{x}$	$\dfrac{\mathsf{M}t}{e}$
no linguist	$\textbf{no.ling} := \dfrac{\textbf{not}\,(\textbf{a.ling}_x \multimap [\ ])}{x}$	$\dfrac{\mathsf{M}t}{e}$

Table 3.1: A sampling of basic meanings. The meanings exhibit the three grades of monadic involvement: the static grammar (*John*, *left*, etc.), the underlying State.Set monad (*a linguist*, *she*), and the continuized tier (*every linguist*, *no linguist*). In addition, we have functions over monadic meanings (*not*, *if*).

the verb, "applying Lift" meaning applying η and then $\uparrow$.[6] In each case we elect to set the result type to $\mathsf{M}t$ when we apply Lift. Since neither the subject nor the verb have scopal effects, the top level of the tower that's derived after forward and backward scopal functional application transparently reflects the scopal effects associated with *a linguist*. Lowering the derived sentential tower yields the expected nondeterministic dynamic proposition, type $\mathsf{M}t$. (After combination I'll often include a list that uses the combined meaning to derive other meanings by applying various rules. For instance, here I derive an $\mathsf{M}t$ program by Lowering the derived two-level tower.)

(3.23)



6 In what follows, whenever I talk about Lifting something, I'll either be referring to an application of $\uparrow$ or to an application of η and $\uparrow$. Which one I mean will always be obvious from the context/towers.

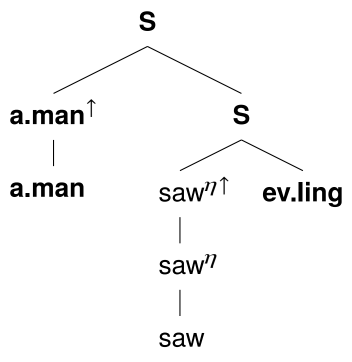
$$\left(\begin{array}{c} \dfrac{\mathsf{M}t}{e} \\[2ex] \text{John} \\[2ex] \dfrac{[\;]}{\mathsf{j}} \end{array} \left(\begin{array}{cc} \dfrac{\mathsf{M}t}{e \to e \to t} & \dfrac{\mathsf{M}t}{e} \\[2ex] \text{saw} & \text{a linguist} \\[2ex] \dfrac{[\;]}{\mathsf{saw}} & \dfrac{\mathbf{a.ling}_x \multimap [\;]}{x} \end{array}\right)\right) = \begin{array}{c} \dfrac{\mathsf{M}t}{t} \\[2ex] \text{John saw a linguist} \\[2ex] \dfrac{\mathbf{a.ling}_x \multimap [\;]}{\mathsf{saw}\, x\, \mathsf{j}} \end{array}$$

$$\rightsquigarrow \quad \mathbf{a.ling}_x \multimap (\mathsf{saw}\, x\, \mathsf{j})^{\eta} \quad \downarrow$$

An analogous derivation goes through for *John saw every linguist*. After combination and Lowering, we end up with the deterministic dynamic proposition $(\forall x.\, \mathsf{ling}\, x \Rightarrow \mathsf{saw}\, x\, \mathsf{j})^{\eta}$.

Derivations with more than one quantifier go just as smoothly. Familiarly by now, the linear bias in the rules for scopal functional application gives rise to linear scopings. An example derivation for the sentence *a man saw every linguist* is given in (3.24). First, we apply Lift to *saw*. Two subsequent applications of scopal functional application yield a tower where the quantificational effects associated with *every linguist* end up trapped within the scope of the nondeterminism introduced by *a man*. Lowering yields the expected nondeterministic dynamic proposition (type $\mathsf{M}t$) about a nondeterministic man seeing every linguist.

(3.24)



$$\left(\begin{array}{c} \dfrac{\mathsf{M}t}{e} \\[2ex] \text{a man} \\[2ex] \dfrac{\mathbf{a.man}_x \multimap [\;]}{x} \end{array} \left(\begin{array}{cc} \dfrac{\mathsf{M}t}{e \to e \to t} & \dfrac{\mathsf{M}t}{e} \\[2ex] \text{saw} & \text{every linguist} \\[2ex] \dfrac{[\;]}{\mathsf{saw}} & \dfrac{\mathbf{ev.ling}\,(\lambda y.\, [\;])}{y} \end{array}\right)\right) = \begin{array}{c} \dfrac{\mathsf{M}t}{t} \\[2ex] \text{a man saw every linguist} \\[2ex] \dfrac{\mathbf{a.man}_x \multimap \mathbf{ev.ling}\,(\lambda y.\, [\;])}{\mathsf{saw}\, y\, x} \end{array}$$

$$\begin{aligned} &\rightsquigarrow \quad \mathbf{a.man}_x \multimap \mathbf{ev.ling}\,(\lambda y.\, (\mathsf{saw}\, y\, x)^{\eta}) \quad \downarrow \\ &= \quad \mathbf{a.man}_x \multimap (\forall y.\, \mathsf{ling}\, y \Rightarrow \mathsf{saw}\, y\, x)^{\eta} \quad \mathbf{ev.ling} \end{aligned}$$

The rules for the combination of two-level towers do not allow for scope ambiguity. In fact, inverse-scope derivations are already present in the grammar. They involve *higher-order continuations* and higher-order scopal functional application. We'll see how this works in Section 3.6.2.

3.6 Extensions

3.6.1 Binding

So far the anaphoric half of the State.Set monad has been an idle bystander. Nothing's been added to the stack, and nothing's depended on the stack for its meaning. A *bind* type-shifter is given in Definition 3.16 and the towers version in Fact 3.15. The Bind type-shifter takes a tower as input and outputs a new tower which pushes a new discourse referent onto the stack (recall that $\widehat{sa}$ updates a stack s by placing a new dref at the end of s). Notice how replacing '$\widehat{sa}$' in (3.16) with 's' yields an expression equivalent to m (i.e. $\lambda k.\, m\, (\lambda as.\, k\; a\; s) = \lambda k.\, m\, k = m$): the *only* effect of ► is adding a dref to the stack.

Definition 3.16 (Bind).

$$m^{\blacktriangleright} := \lambda k.\, m\, (\lambda as.\, k\; a\; \widehat{sa})$$

Fact 3.15 (Bind in towers).

$$\left(\begin{array}{c} \dfrac{\mathsf{M}\beta}{\alpha} \\ \\ exp \\ \\ \dfrac{f\,[m_v \multimap [\,]]}{v} \end{array}\right)^{\blacktriangleright} = \begin{array}{c} \dfrac{\mathsf{M}\beta}{\alpha} \\ \\ exp \\ \\ \dfrac{f\,[m_v^{\triangleright} \multimap [\,]]}{v} \end{array}$$

Proof.

$$\begin{aligned} \left(\frac{f\,[m_v \multimap [\,]]}{v}\right)^{\blacktriangleright} &= \lambda k.\, f\,[m_v \multimap (\lambda as.\, k\; a\; \widehat{sa})\; v] && \blacktriangleright \\ &= \lambda k.\, f\,[m_v \multimap \lambda s.\, k\; v\; \widehat{sv}] && = \\ &= \lambda k.\, f\,[m_v^{\triangleright} \multimap k\; v] && \triangleright \\ &= \frac{f\,[m_v^{\triangleright} \multimap [\,]]}{v} && = \end{aligned}$$

□

Some examples of bind-shifts—for *John*, *a linguist*, and *every linguist*—are given in Fact 3.16. The effect of each shift is to introduce a dref onto the scopal tier.

Fact 3.16 (Examples of bind-shifts).

$$\mathsf{j}^{\eta\uparrow\blacktriangleright} = \left(\frac{\mathsf{j}^{\eta}_{v} \multimap [\,]}{v}\right)^{\blacktriangleright} = \frac{\mathsf{j}^{\eta\triangleright}_{v} \multimap [\,]}{v}$$

$$\textbf{a.man}^{\uparrow\blacktriangleright} = \left(\frac{\textbf{a.man}_{v} \multimap [\,]}{v}\right)^{\blacktriangleright} = \frac{\textbf{a.man}^{\triangleright}_{v} \multimap [\,]}{v}$$

$$\left(\frac{\textbf{ev.ling}\,(\lambda x.\,[\,])}{x}\right)^{\blacktriangleright} = \left(\frac{\textbf{ev.ling}\,(\lambda x.\,x^{\eta}_{v} \multimap [\,])}{v}\right)^{\blacktriangleright} = \frac{\textbf{ev.ling}\,(\lambda x.\,x^{\eta\triangleright}_{v} \multimap [\,])}{v}$$

Two signposts: The bind type-shifter is naturally polymorphic, though in this chapter I'll only be applying it to continuized DPs. In addition, the bind-shifter can apply to pronouns and pronoun-containing constituents just as well as it applies to DPs like *Bill* or *a man*. That is, pronouns (and things that contain pronouns) can make drefs. I will explore some welcome consequences of bind-polymorphism and bind-shifted pronominal constituents in Chapter 5.

Before moving to some derivations, I'll mention a couple of notable convergences this semantics has with the non-monadic DyS semantics from Chapter 2. First, the result of bind-shifting a Lifted indefinite turns out to be the DyS semantics for the indefinite (though with a different type; $\uparrow$/$\multimap$ presupposes that the computation will yield objects in the State.Set monad):

Fact 3.17 (DyS correspondence #1).

$$\begin{aligned}
\textbf{a.man}^{\uparrow\blacktriangleright} &= \lambda k.\,\textbf{a.man}^{\triangleright}_{v} \multimap k\,v && \uparrow\text{, Fact 3.16}\\
&= \lambda k.\,\textbf{a.man}_{v} \multimap \lambda s.\,k\,v\,\widehat{sv} && \triangleright\\
&= \lambda ks.\bigcup_{\mathsf{man}\,v} k\,v\,\widehat{sv} && \textbf{a.man}, \multimap
\end{aligned}$$

Second, Lifting a pronoun into a scope-taker results in essentially the DyS semantics for pronouns (again, modulo the types). As with indefinites, we've in a sense derived the DyS semantics from the underlying monadic lexical entries plus scope (see Table 3.1 a couple pages back for a refresher on the semantics of pronouns):

Fact 3.18 (DyS correspondence #2).

$$\begin{aligned}
\textbf{pro}^{\uparrow} &= \lambda k.\,\textbf{pro}_{v} \multimap k\,v && \uparrow\\
&= \lambda ks.\,k\,s_{\top}\,s && \textbf{pro}, \multimap
\end{aligned}$$

Onto some illustrations of how binding works. A binding interpretation for *John$_i$ rubbed his$_i$ head* is derived in (3.25). As in Chapter 2, I treat the relational noun head as an $e \to e$ function that maps every individual to their head and act as if the semantics of the genitive *his* is no different from the semantics of *he*. I've Lifted *John*, *rubbed*, *his*, and *head*, crucially applied the bind type-shifter to *John*, and fixed all of our above-the-line types to $\mathsf{M}t$. The derivation proceeds via forward and backward scopal functional application. This brings the pronoun within the (immediate) scope of the bind-shifted *John*, and binding is achieved:

(3.25)
$$\left(\begin{array}{c} \dfrac{\mathsf{M}t}{e} \\[1ex] \text{John} \\[1ex] \dfrac{\mathsf{j}_x^{\eta\triangleright} \multimap [\,]}{x} \end{array} \left(\begin{array}{c} \dfrac{\mathsf{M}t}{e \to e \to t} \\[1ex] \text{rubbed} \\[1ex] \dfrac{[\,]}{\mathsf{rubbed}} \end{array} \left(\begin{array}{cc} \dfrac{\mathsf{M}t}{e} & \dfrac{\mathsf{M}t}{e \to e} \\[1ex] \text{his} & \text{head} \\[1ex] \dfrac{\mathbf{pro}_y \multimap [\,]}{y} & \dfrac{[\,]}{\mathsf{head}} \end{array}\right)\right)\right) = \begin{array}{c} \dfrac{\mathsf{M}t}{t} \\[1ex] \text{John rubbed his head} \\[1ex] \dfrac{\mathsf{j}_x^{\eta\triangleright} \multimap \mathbf{pro}_y \multimap [\,]}{\mathsf{rubbed}\,(\mathsf{head}\,y)\,x} \end{array}$$

$$\begin{array}{rrl} \rightsquigarrow & \mathsf{j}_x^{\eta\triangleright} \multimap \mathbf{pro}_y \multimap (\mathsf{rubbed}\,(\mathsf{head}\,y)\,x)^{\eta} & \downarrow \\ = & \mathsf{j}_x^{\eta\triangleright} \multimap (\mathsf{rubbed}\,(\mathsf{head}\,x)\,x)^{\eta} & \text{Binding} \\ = & \lambda s.\,\{\langle \mathsf{rubbed}\,(\mathsf{head}\,\mathsf{j})\,\mathsf{j},\ \widehat{s\mathsf{j}}\rangle\} & \eta,\ \triangleright,\ \multimap \end{array}$$

An analogous derivation for *a man*$_i$ *rubbed his*$_i$ *head* yields the nondeterministic dynamic proposition $\mathbf{a.man}_x \multimap (\mathsf{rubbed}\,(\mathsf{head}\,x)\,x)^{\eta}$ after combination and Lowering, as expected. *Every man*$_i$ *rubbed his*$_i$ *head* is likewise associated with a bound interpretation; there, the result after Lowering is equivalent to a trivial $\mathsf{M}t$ program, namely $(\forall x.\,\mathsf{man}\,x \Rightarrow \mathsf{rubbed}\,(\mathsf{head}\,x)\,x)^{\eta}$. As was the case for the State and State.Set monads in Chapter 2, there is no need in any of these cases to co-index antecedent and pronoun; the antecedent simply plonks a dref onto the stack, and the pronoun simply grabs the most recently introduced dref (as there, we can enrich the semantics of pronouns to grab drefs from arbitrary stack positions, still without requiring binder/bind-ee co-indexation).

Because dref-hosting programs live on the scopal tier, where linear precedence, rather than surface c-command, is the order of the day, an absence of surface c-command does not disrupt binding (cf. e.g. Barss & Lasnik 1986, Larson 1988). See (3.26) (an analogous property inheres in the grammars of Shan & Barker 2006, Barker & Shan 2008, 2014):

(3.26)
$$\left(\left(\begin{array}{cc} \dfrac{\mathsf{M}t}{e} & \dfrac{\mathsf{M}t}{e \to e} \\[1ex] \text{John's} & \text{mom} \\[1ex] \dfrac{\mathsf{j}_x^{\eta\triangleright} \multimap [\,]}{x} & \dfrac{[\,]}{\mathsf{mom}} \end{array}\right)\left(\begin{array}{cc} \dfrac{\mathsf{M}t}{e \to e \to t} & \dfrac{\mathsf{M}t}{e} \\[1ex] \text{saw} & \text{him} \\[1ex] \dfrac{[\,]}{\mathsf{saw}} & \dfrac{\mathbf{pro}_y \multimap [\,]}{y} \end{array}\right)\right) = \begin{array}{c} \dfrac{\mathsf{M}t}{t} \\[1ex] \text{John's mom saw him} \\[1ex] \dfrac{\mathsf{j}_x^{\eta\triangleright} \multimap \mathbf{pro}_y \multimap [\,]}{\mathsf{saw}\,y\,(\mathsf{mom}\,x)} \end{array}$$

$$\begin{array}{rrl} \rightsquigarrow & \mathsf{j}_x^{\eta\triangleright} \multimap \mathbf{pro}_y \multimap (\mathsf{saw}\,y\,(\mathsf{mom}\,x))^{\eta} & \downarrow \\ = & \mathsf{j}_x^{\eta\triangleright} \multimap (\mathsf{saw}\,x\,(\mathsf{mom}\,x))^{\eta} & \text{Binding} \\ = & \lambda s.\,\{\langle \mathsf{saw}\,\mathsf{j}\,(\mathsf{mom}\,\mathsf{j}),\ \widehat{s\mathsf{j}}\rangle\} & \eta,\ \triangleright,\ \multimap \end{array}$$

Analogous derivations go through for *a man$_i$'s mom saw him$_i$* and *every man$_i$'s mom saw him$_i$*. For the former, we derive $\textbf{a.man}_x \multimap (\mathsf{saw}\,x\,(\mathsf{mom}\,x))^\eta$. For the latter, we end up with a pure computation, namely $(\forall x.\,\mathsf{man}\,x \Rightarrow \mathsf{saw}\,x\,(\mathsf{mom}\,x))^\eta$.

3.6.2 Inverse scope

Inverse-scope derivations are already present in this grammar. As a first step, notice that we can apply Lift *to a tower*, as in Fact 3.19. We'll call Lifting a tower in this way *external* Lift. External Lift puts us in the presence of *higher-order continuations*: as the linear semantics $\lambda c.\,c\,(\lambda k.\,f\,[k\,a])$ makes clear, this tower's continuation c is *itself* a function over towers.

Fact 3.19 (Externally Lifting a tower). Fixing some γ:

$$\left(\begin{array}{c}\mathsf{M}\beta\\ \hline \alpha\\ \\ exp\\ \\ f\,[\,]\\ \hline a\end{array}\right)^{\eta\uparrow} = \begin{array}{c}\mathsf{M}\gamma\\ \hline \mathsf{M}\beta\\ \hline \alpha\\ \\ exp\\ \\ [\,]\\ \hline f\,[\,]\\ \hline a\end{array}$$

We can also apply Lift *inside* a tower, as in Definition 3.17 (cf. Shan & Barker 2006, Barker & Shan 2008, 2014).[7] We'll call this variety of Lifting *internal* Lift. Internal Lift turns a two-level tower into a three-level tower by inserting a new scopal tier in the middle. Anything on the second floor of the pre-Lift two-level tower finds itself on top of the post-Lift three-level tower.

Definition 3.17 (Internal Lift).

$$m^{\upuparrows} := \lambda c.\,m\,(\lambda v.\,c\,(\lambda k.\,k\,v))$$

Fact 3.20 (Internal Lift in tower notation). Fixing some γ:

$$\left(\begin{array}{c}\mathsf{M}\beta\\ \hline \alpha\\ \\ exp\\ \\ f\,[\,]\\ \hline a\end{array}\right)^{\upuparrows} = \begin{array}{c}\mathsf{M}\beta\\ \hline \mathsf{M}\gamma\\ \hline \alpha\\ \\ exp\\ \\ f\,[\,]\\ \hline [\,]\\ \hline a\end{array}$$

7 See Appendix A for a derivation of internal Lift and all the combinators to follow from $\mathbf{S}$, η, and $\multimap$.

As you might expect, effects on higher levels will out-scope those on lower levels, in spite of the grammatical preference for left-to-right evaluation: as the semantics in Definition 3.17 makes clear, internal Lift places the new continuation *c within the scope* of *m*; in Fact 3.19, by contrast, *m* finds itself within the scope of the new continuation.

Example (3.27) applies internal Lift to *every linguist* (fixing γ to t). We create a new second story in the tower, moving all the scopal action onto the third level. The result is equivalent to $\lambda c.\, \textbf{ev.ling}\, (\lambda y.\, c\, (\lambda k.\, k\, y))$, i.e. with **ev.ling** scoping over the higher-order continuation c.

$$(3.27)\qquad \left(\begin{array}{c} \begin{array}{c} \mathsf{M}t \\ \hline e \end{array} \\[2ex] \text{every linguist} \\[2ex] \begin{array}{c} \textbf{ev.ling}\,(\lambda y.\,[\,]) \\ \hline y \end{array} \end{array}\right)^{\upuparrows} = \begin{array}{c} \begin{array}{c} \mathsf{M}t \\ \hline \mathsf{M}t \\ \hline e \end{array} \\[2ex] \text{every linguist} \\[2ex] \begin{array}{c} \textbf{ev.ling}\,(\lambda y.\,[\,]) \\ \hline [\,] \\ \hline y \end{array} \end{array}$$

An operation for combining three-level towers is also present in the grammar (in fact, so are operations for combining towers of arbitrary heights). See Definition 3.18. Three-level combination scopes the top levels of two three level towers in their surface order and does scopal functional application on the bottom two stories. The upshot for the tower notation, given in Fact 3.21, is very much in line with how two-level towers were combined: on the bottom-most level, we do regular functional application. On the upper tiers, we compose the scopal effects in their surface order (and require matching types on each level). Notice in particular that *h* out-scopes *g* post-combination, despite the fact that *h* is associated with *right-exp*, and *g* is associated with *left-exp*.

Definition 3.18 (Three-level combination).

$$\begin{aligned} \mathbb{S}\, M\, N :=\ & \lambda c.\, M\, (\lambda m. \\ & \quad N\, (\lambda n. \\ & \qquad c\, (\mathbf{S}\, m\, n))) \end{aligned}$$

Fact 3.21 (Three-level tower combination). $\{\gamma,\ \delta\} := \{\alpha,\ \alpha \to \beta\}$, for some types α and β:

$$\left(\begin{array}{cc} \begin{array}{c} \sigma \\ \hline \rho \\ \hline \gamma \end{array} & \begin{array}{c} \sigma \\ \hline \rho \\ \hline \delta \end{array} \\[2ex] \textit{left-exp} & \textit{right-exp} \\[2ex] \begin{array}{c} f\,[\,] \\ \hline g\,[\,] \\ \hline x \end{array} & \begin{array}{c} h\,[\,] \\ \hline i\,[\,] \\ \hline y \end{array} \end{array}\right) = \begin{array}{c} \begin{array}{c} \sigma \\ \hline \rho \\ \hline \beta \end{array} \\[2ex] \textit{left-exp right-exp} \\[2ex] \begin{array}{c} f\,[h\,[\,]] \\ \hline g\,[i\,[\,]] \\ \hline \mathsf{A}\, x\, y \end{array} \end{array}$$

What goes up must eventually come down. There's two ways to Lower a three-level tower. One way is to simply apply Lower to the *bottom* two stories of the tower to yield a two-story tower whose bottom two floors have been collapsed (we'll see examples of this in Chapter 4). More relevant at the moment is a version of Lower which evaluates a three-level tower in one fell swoop. The semantics of the one-fell-swoop Lower is, fittingly, the Montague-lift of regular Lower. See Definition 3.19. This Lowering operation collapses three levels into one, while turning the bottom level into a trivial program. See Fact 3.22 for the towers version. The only difference from the two-level Lower operation is that we discharge two side levels of side effects instead of just one.

Definition 3.19 (Three-level Lower).

$$M^{\downdownarrows} := M\,(\lambda m.\, m\,\eta)$$

Fact 3.22 (Three-level Lower in towers).

$$\left(\begin{array}{c} \begin{array}{c} \mathsf{M}\alpha \\ \hline \mathsf{M}\alpha \\ \hline \alpha \end{array} \\[2ex] exp \\[2ex] \begin{array}{c} g\,[\,] \\ \hline h\,[\,] \\ \hline a \end{array} \end{array}\right)^{\downdownarrows} = \begin{array}{c} \mathsf{M}\alpha \\[2ex] exp \\[2ex] g\,[h\,[a^{\eta}]] \end{array}$$

Here's an inverse-scope derivation for *a man saw every linguist*. I applied external Lift to *a man* and internal Lift to *every linguist*. This means *every linguist* out-scopes *a man*. I also Lifted *saw* twice and bind-shifted the subject DP so that we can observe the effect of wide-scoping a dynamically closed operator on anaphoric potential post-evaluation. Combination is via two instances of (higher-order) scopal functional application. We end the derivation by Lowering the composite three-story tower into a deterministic dynamic proposition.

(3.28)
$$\left(\begin{array}{c} \begin{array}{c} \mathsf{M}t \\ \hline \mathsf{M}t \\ \hline e \end{array} \\[2ex] \text{a man} \\[2ex] \begin{array}{c} [\,] \\ \hline \textbf{a.man}^{\triangleright}_{x} \multimap [\,] \\ \hline x \end{array} \end{array} \left(\begin{array}{cc} \begin{array}{c} \mathsf{M}t \\ \hline \mathsf{M}t \\ \hline e \to e \to t \end{array} & \begin{array}{c} \mathsf{M}t \\ \hline \mathsf{M}t \\ \hline e \end{array} \\[2ex] \text{saw} & \text{every linguist} \\[2ex] \begin{array}{c} [\,] \\ \hline [\,] \\ \hline \mathsf{saw} \end{array} & \begin{array}{c} \textbf{ev.ling}\,(\lambda y.\,[\,]) \\ \hline [\,] \\ \hline y \end{array} \end{array}\right)\right) = \begin{array}{c} \begin{array}{c} \mathsf{M}t \\ \hline \mathsf{M}t \\ \hline t \end{array} \\[2ex] \text{a man saw every linguist} \\[2ex] \begin{array}{c} \textbf{ev.ling}\,(\lambda y.\,[\,]) \\ \hline \textbf{a.man}^{\triangleright}_{x} \multimap [\,] \\ \hline \mathsf{saw}\,y\,x \end{array} \end{array}$$

$$
\begin{array}{lll}
\leadsto & \textbf{ev.ling}\,(\lambda y.\,\textbf{a.man}_x \multimap (\mathsf{saw}\,y\,x)^\eta) & \Downarrow \\
= & \textbf{ev.ling}\,(\lambda ys.\,\{\langle \mathsf{saw}\,y\,x,\ \widehat{sx}\rangle \mid \mathsf{man}\,x\}) & \eta, \multimap, \rhd, \textbf{a.man} \\
= & (\forall y.\,\mathsf{ling}\,y \Rightarrow \exists x.\,\mathsf{man}\,x \wedge \mathsf{saw}\,y\,x)^\eta & \textbf{ev.ling}
\end{array}
$$

The truth condition is as desired: every linguist was seen by some man or other (notice for the last step that **ev.ling** requires that for every linguist y, there is some True boolean among the values saw $y\ x$ (for x some man). This will be so just in case, for every linguist y, there is a man x such that x saw y: the nondeterminism created by *a man* is quashed by the existential closure over outputs that inheres in the semantics of **ev.ling**. Importantly, even though we have applied the bind type-shifter to *a man*, giving the dynamically closed *every linguist* widest scope renders this moot. All the nondeterminism and novel binding information generated in the scope of *every linguist* is used only to calculate truth conditions and is then promptly discarded.

3.6.3 Composing DPs

The last extension we'll look at is the composition of complex DPs like *a linguist* and *every owl that Al saw*. This goal of this section is to give a semantics for determiners and to show that a compositional treatment of complex DPs follows from the grammar we've been using, along with some natural lexical entries for extraction gaps and relative pronouns. It isn't my aim to give an in-depth treatment of the syntax or semantics of determiners, relative clauses, or DP-internal composition. I will give enough detail to make it plausible that my semantics has no difficulty assembling complex DPs, to highlight some points of interest for the characterization of the Continuation monad, and to give a basis for the discussion of exceptional scope out of relative clauses in Chapter 4.

We begin by defining a lexical entry for the indefinite determiner *a* and using that to define lexical entries for *every* and *no*. See Definition 3.20. The meaning assigned to the indefinite determiner (**a**) is a function, type $(e \to \mathsf{M}t) \to \mathsf{M}e$, from dynamic properties c into monadic programs bottling up the individuals x who "satisfy" c (while retaining the updated stacks generated for each satisfying individual). By exploiting this entry and the lexical semantics for negation motivated at the beginning of in this chapter, we can give a semantics for *every* (**ev**) and *no* (**no**). The entry for *every* draws on a standard first-order equivalence—viz. $\forall x.\,P\,x \Rightarrow Q\,x$ iff $\neg\exists x.\,P\,x \wedge \neg Q\,x$, and the entry for *no* simply omits a negation from *every*.

Definition 3.20 (Determiner semantics).

Abbreviated	Expanded	Type
a	$\lambda cs.\,\{\langle x,\ s'\rangle \mid \langle \mathsf{True},\ s'\rangle \in c\,x\,s\}$	$(e \to \mathsf{M}t) \to \mathsf{M}e$
ev	$\lambda ck.\,\textbf{not}\,((\textbf{a}\,c)_x \multimap \textbf{not}\,(k\,x))$	$(e \to \mathsf{M}t) \to (e \to \mathsf{M}t) \to \mathsf{M}t$
no	$\lambda ck.\,\textbf{not}\,((\textbf{a}\,c)_x \multimap k\,x)$	$(e \to \mathsf{M}t) \to (e \to \mathsf{M}t) \to \mathsf{M}t$

I wish to suggest that determiners interact with their restrictors by *taking scope* (this suggestion is by no means original to me; see e.g. Heim 1982, Barker 1995, Heim 1997 for pertinent discussion), and that this possibility *follows* from a harmless generalization of the notion of a continuized type. Notice as a first step that each determiner's type has the form $(e \to \mathsf{M}t) \to \alpha$, for some α. This is quite reminiscent of, though more general than, the continuized type schemas we have been working with, which have the form $(\alpha \to \mathsf{M}\beta) \to \mathsf{M}\beta$. Allow me to take a brief theoretical digression to show how types of this form can be brought into the scope-taking fold (readers familiar with the systems of Shan & Barker 2006, Barker & Shan 2008, 2014 can skip straight head to the analysis.)

Following Shan & Barker 2006, Barker & Shan 2008, 2014, we can represent any type of the form $(\alpha \to \rho) \to \sigma$ in terms of a *tripartite* tower, as in Definition 3.21. A tower of this form can be pronounced, "contributes a local argument of type α, takes scope over something with type ρ, and returns something with type σ". This represents a generalization of the bipartite towers we've been using so far, in the specific sense that bipartite towers can be understood as abbreviations for tripartite towers. See Definition 3.22.

Definition 3.21 (Types).

$$\frac{\sigma \mid \rho}{\alpha} \;:=\; (\alpha \to \rho) \to \sigma$$

Definition 3.22 (Bipartite abbreviations for tripartite types).

$$\frac{\rho}{\alpha} \;:=\; \frac{\rho \mid \rho}{\alpha}$$

Wadler 1994 emphasizes that type schemas of this form imply a construct for scopal program composition *more general* than a proper monad. Importantly, however, the necessary generalization is *confined to the types*: the semantics of injection and sequencing remain unchanged, and the more general construct—call it the Continuation monad$^+$—still obeys all the monad laws.

Thus, the semantics of tripartite tower combination is unchanged from before, though we must be a bit more specific about the types (Shan & Barker 2006, Barker & Shan 2008, 2014). See Fact 3.23. While the lower level remains unchanged from before, on the upper level we now require matching *adjacent types*: the type returned by the expression on the right has to match the result type expected by the expression on the left (here, ρ). Matching adjacent types cancel out on combination.

Fact 3.23 (Two-level combination in the Continuation monad$^+$). $\{\gamma, \delta\} := \{\alpha, \alpha \to \beta\}$

$$\left(\begin{array}{cc} \dfrac{\sigma \mid \rho}{\gamma} & \dfrac{\rho \mid \tau}{\delta} \\ \\ \textit{left-exp} & \textit{right-exp} \\ \\ \dfrac{f\,[\,]}{x} & \dfrac{g\,[\,]}{y} \end{array}\right) \;:=\; \begin{array}{c} \dfrac{\sigma \mid \tau}{\beta} \\ \\ \textit{left-exp right-exp} \\ \\ \dfrac{f\,[g\,[\,]]}{\mathsf{A}\,x\,y} \end{array}$$

Likewise, the semantics of Lower is unchanged from before (i.e. as before, it involves application to a trivial continuation $k_0 \coloneqq \eta$). Once again, the difference from the earlier version is confined to the types. Here we require the lower type to be α and the upper right type to be $\mathsf{M}\alpha$ (for any α).

Fact 3.24 (Lower in the Continuation monad$^+$).

$$\left(\begin{array}{c} \dfrac{\beta \,\big|\, \mathsf{M}\alpha}{\alpha} \\[1em] \mathit{exp} \\[1em] \dfrac{f\,[\;]}{a} \end{array}\right)^{\downarrow} = \begin{array}{c} \beta \\[1em] \mathit{exp} \\[1em] f\,[a^{\eta}] \end{array}$$

These theoretical preliminaries out of the way, we can now observe that the lexical entries for determiners we gave at the outset can be conceived of as scope-takers, in terms of tripartite towers (and bipartite abbreviations). These towers rely on a routine λ-theoretic equivalence, for example that $\textbf{ev} = \lambda c.\, \textbf{ev}\, c = \lambda c.\, \textbf{ev}\,(\lambda x.\, c\, x)$. In other words, we treat the c argument as a continuation/scope. The reader can check that the tower type for each of these meanings is equivalent to its linear type.

Fact 3.25 (Determiners as tripartite towers).

$$\begin{array}{ccc} \dfrac{\mathsf{M}e \,\big|\, \mathsf{M}t}{e} & \dfrac{\frac{\mathsf{M}t}{e} \,\Big|\, \mathsf{M}t}{e} & \dfrac{\frac{\mathsf{M}t}{e} \,\Big|\, \mathsf{M}t}{e} \\[1.5em] \text{a} & \text{every} & \text{no} \\[1.5em] \dfrac{\textbf{a}\,(\lambda x.\,[\;])}{x} & \dfrac{\textbf{ev}\,(\lambda x.\,[\;])}{x} & \dfrac{\textbf{no}\,(\lambda x.\,[\;])}{x} \end{array}$$

And that is all we need to compose up a complex DP like *every owl*. See (3.29). After Lifting *owl*, we combine it with the determiner by scopal application. This brings us to the tower on the right, which matches the type schema for tripartite Lower. After applying Lower, we are left with the expected bipartite tower: the result is equivalent to $\lambda k.\, \textbf{ev}\,(\lambda x.\,(\mathsf{owl}\, x)^{\eta})\,(\lambda x.\, k\, x)$, which is equivalent to $\lambda k s.\, \{\langle \forall x.\, \mathsf{owl}\, x \Rightarrow \exists s'.\, \langle \mathsf{True},\, s' \rangle \in k\, x\, s,\ s \rangle\}$

(3.29)
$$\left(\begin{array}{cc} \dfrac{\frac{\mathsf{M}t}{e} \,\Big|\, \mathsf{M}t}{e} & \dfrac{\mathsf{M}t}{e \to t} \\[1.5em] \text{every} & \text{owl} \\[1.5em] \dfrac{\textbf{ev}\,(\lambda x.\,[\;])}{x} & \dfrac{[\;]}{\mathsf{owl}} \end{array}\right) = \begin{array}{c} \dfrac{\frac{\mathsf{M}t}{e} \,\Big|\, \mathsf{M}t}{t} \\[1.5em] \text{every owl} \\[1.5em] \dfrac{\textbf{ev}\,(\lambda x.\,[\;])}{\mathsf{owl}\, x} \end{array}$$

$$\leadsto \underbrace{\textbf{ev}\,(\lambda x.\,(\mathsf{owl}\ x)^{\eta})}_{\frac{\mathsf{M}t}{e}} \ \downarrow$$

The final piece is a treatment of relative clauses like *that Al saw*. The crucial pieces for building a relative clause are a meaning for an extraction gap and a meaning for the relative pronoun *that*. I will assume, with much of the literature (e.g. Heim & Kratzer 1998), that a gap has the same sort of denotation as a pronominal element.[8] As for the semantics of the relative pronoun, I will assume that it is synonymous with *and*. (Neither of these assumptions is crucial for an account of complex DPs or the analyses in later chapters. I choose them for concreteness and simplicity.)

Definition 3.23 (Gaps and relative pronouns).

$$[\![\textsc{gap}]\!] := \textbf{pro} \qquad [\![\text{that}]\!] := (\wedge)$$

Using these pieces, we can derive *every owl that Al saw*. See (3.30).[9] Notice that my bracketing is characteristic of an "NP-S" (in more modern terms, DP-S) analysis of the complex DP (e.g. Bach & Cooper 1978). That is, *every owl* forms a constituent to the exclusion of the relative clause. (A Det-Nom syntax, i.e. where *owl* and the relative clause form a constituent to the exclusion of *every*, is possible but requires a different semantics for relative pronouns and/or gaps). Notice also that since the gap requires an antecedent, I have Bind-shifted *every* before composing it with *owl*.

(3.30)
$$\left(\begin{array}{c} \begin{array}{c|c} \frac{\mathsf{M}t}{e} & \mathsf{M}t \\ \hline \multicolumn{2}{c}{t} \end{array} \\ \text{every owl} \\ \frac{\textbf{ev}\,(\lambda x.\, x_v^{\eta\rhd} \multimap [\,])}{\mathsf{owl}\ v} \end{array} \left(\begin{array}{cc} \frac{\mathsf{M}t}{t \to t \to t} & \frac{\mathsf{M}t}{t} \\ \text{that} & \text{Al saw } \textsc{gap} \\ \frac{[\,]}{(\wedge)} & \frac{\textbf{pro}_y \multimap [\,]}{\mathsf{saw}\ y\ \mathsf{a}} \end{array}\right)\right) = \begin{array}{c} \begin{array}{c|c} \frac{\mathsf{M}t}{e} & \mathsf{M}t \\ \hline \multicolumn{2}{c}{t} \end{array} \\ \text{every owl that Al saw } \textsc{gap} \\ \frac{\textbf{ev}\,(\lambda x.\, x_v^{\eta\rhd} \multimap \textbf{pro}_y \multimap [\,])}{\mathsf{owl}\ v \wedge \mathsf{saw}\ y\ \mathsf{a}} \end{array}$$

$$\begin{aligned} &\leadsto \textbf{ev}\,(\lambda x.\, x_v^{\eta\rhd} \multimap \textbf{pro}_y \multimap (\mathsf{owl}\ v \wedge \mathsf{saw}\ y\ \mathsf{a})^{\eta}) \ \downarrow \\ &= \underbrace{\textbf{ev}\,(\lambda x.\, x_v^{\eta\rhd} \multimap (\mathsf{owl}\ v \wedge \mathsf{saw}\ v\ \mathsf{a})^{\eta})}_{\frac{\mathsf{M}t}{e}} \quad \text{Binding} \end{aligned}$$

Combination brings **pro** within the immediate scope of the dref contributed by *every*, and binding of the gap is achieved. After an application of Lower, we are left with the expected bipartite tower. The resulting meaning expands into $\lambda ks.\,\{\langle \forall v.\,(\mathsf{owl}\ v \wedge \mathsf{saw}\ v\ \mathsf{a}) \Rightarrow \exists s'.\,\langle \mathsf{True},\ s' \rangle \in k\ v\ \widehat{sv},\ s\rangle\}$.

And that, in a nutshell, is my treatment of complex DPs. Though I do not provide detailed

8 Of course, we will have to constrain what a gap can be bound by to avoid rank over-generation, but that is a matter for a more worked out syntax than what I can propose here. See Heim & Kratzer 1998: 123–128 for discussion.

9 The analysis in (3.30) elides one important fact: relative clauses are scope islands. We return to this in Section 4.5.3.

analyses here, the reader can verify that the analysis readily accommodates stacked relative clauses, as in constructions like *every owl that Al saw that hooted*, as well as donkey anaphora out of DP, as in *every boy who saw an owl*$_i$ *hooted at it*$_i$.

3.7 Conclusion

This chapter added dynamically closed operators, quantifiers, and scope-taking to the State.Set semantics developed in Chapter 2. The result is a fragment that offers robust treatments of quantificational scope (both surface and inverse varieties), binding, and the composition of complex DPs and relative clauses. I drew out a connection between scope-taking and what I called an underlying monad $\mathcal{M}$ by identifying Lift ($\uparrow$) with $\mathcal{M}$'s sequencing operation $\multimap$, and the trivial continuation k_0 on which evaluation/Lower ($\downarrow$) is based with $\mathcal{M}$'s injection function η (and I argued that this perspective represents a generalization of previous continuations-based semantics for natural language). I showed how this setup allows side effects in the underlying monad to interact *scopally* with each other and with irreducibly scopal expressions such as *every linguist*. The next chapter shows that coupling this setup with a natural proposal for scope islands automatically predicts that side effects can take scope beyond the boundaries of a scope island.

In sum, the only type-shifters I rely on are **S**, η, $\multimap$, and ►. It's useful to think about the relation between this apparatus and standard approaches to similar phenomena. Take the textbook semantics of Heim & Kratzer 1998. There is machinery for side effect composition (i.e. the Reader monad underlies SSA), scope (LF and quantifier raising), and binding (assignments and a rule for predicate abstraction). Correspondingly, our grammar recognizes an underlying State.Set monad for side effects, uses **S** (along with the underlying monad's injection and sequencing functions) to handle scope-taking, and effects binding with ►. While the continuations-based approach to scopal composition is certainly different in perspective from LF and QR—i.e. it takes a denotational/semantic rather than operational/syntactic approach to assembling scopal computations—it is not fundamentally different *in kind*: as emphasized by Szabolcsi 2010, 2011 both sorts of approaches deliver the same argument to a scopal expression in the end; they just build that argument in different ways. There is thus a reasonable case to be made that the only true enrichment in the proposed fragment is the presence of the State.Set monad in lieu of Heim & Kratzer's 1998 Reader monad.

Part II

Application: scope-taking after evaluation

Chapter 4

The scope of indefiniteness

4.1 Introduction

This chapter tackles scope islands and the ability of indefinites and disjunctions to scope out of islands. I propose that a scope island is any domain that is *obligatorily evaluated*—alternatively, any domain which is obligatorily Lowered. This corresponds to a semanticization of constraints on scope-taking which are usually stated operationally—i.e. in terms of constraints on QR/grammatical LFs. I detail how the semantics developed in Chapters 2 and 3 immediately predicts that any side effects that survive evaluation (in particular, the nondeterministic and potentially state-changing side effects incurred by indefinites and disjunction) may take scope out of islands, offering a semantic explanation for exceptional scope-taking and a unified perspective on an intricate collection of empirical facts.

4.2 Empirical domain

4.2.1 Exceptionally scoping indefinites

Along with being able to bind pronouns in subsequent sentences, indefinites also readily scope out of scope islands (demarcated with ⟨angled brackets⟩). Sentence (4.1a) can describe a situation in which I've got a single rich relative who's put me in her will (though I may not know who she is). This reading requires the existential quantifier contributed by the indefinite DP *a relative of mine* to have scope over the conditional. As with cross-sentential anaphora, DPs headed by truly quantificational determiners like *every* and *no* lack this flexibility. Sentences (4.1b) and (4.1c) can't be interpreted with the embedded quantifier scoping over the conditional. This is the sense in which the scope of the indefinite in the indicated reading of (4.1a) requires exceptional scope-taking.

(4.1) a. If ⟨a relative of mine dies⟩, I'll inherit a house. ($\exists >$ if)
b. If ⟨every relative of mine dies⟩, I'll inherit a house. ($*\forall >$ if)
c. If ⟨no relative of mine dies⟩, I'll inherit a house. ($*\neg\exists >$ if)

The earliest discussions of exceptionally scoping indefinites in the formal semantics literature are Fodor & Sag 1982 and Farkas 1981. Though Fodor & Sag (along with Heim 1982:223) argue that intermediate exceptional scope is impossible (claiming that exceptionally scoping indefinites are referential and thus scopeless), Farkas persuasively argues otherwise (Farkas's view is the current consensus in the literature). For example, it's possible to interpret (4.2a) in such a way that teachers vary with students but not with books, i.e. as meaning that for no student x was there a teacher of x's y such that x read every book y recommended.

(4.2) a. No$_i$ student read every book that ⟨a teacher of his$_i$ recommended⟩.
b. No$_i$ student read every book that ⟨every teacher of his$_i$ recommended⟩.

This scope-taking is exceptional because the (tensed) relative clause is a scope island: the only possible reading of (4.2b) (pronounced with neutral intonation) has *every teacher of his$_i$* scoping within its minimal relative clause.

4.2.2 Feeding binding

Importantly, exceptional existential scope-takers can bind pronouns. In fact, they do so as if they were actually scoping in their existential scope position: the pronouns anaphoric to exceptionally scoping indefinites necessarily range over individuals that satisfy the indefinite's restrictor (e.g. Abusch 1994, Chierchia 2005). On the indicated exceptional-scope reading, (4.1a) can be continued with *she put me in her will last April*, where *she* necessarily ranges over relatives of mine (whose dying nets me a house). This continuation is infelicitous when the existential scope of the indefinite *a relative of mine* is confined to the antecedent of the conditional. Similar facts hold for (4.3a) and (4.3b): the indicated readings are infelicitous when the superscripted indefinite scopes within its minimal relative clause, but become felicitous when the indefinite takes intermediate or maximal exceptional scope.

(4.3) a. If you manage to read every article ⟨written about a$_i$ famous problem in binding theory⟩, you'll certainly end up an expert on it$_i$.
b. Every student who attended every talk ⟨given at a$_i$ prominent semantics conference⟩ left it$_i$ feeling satisfied.

Note that the intermediate exceptional scope readings have donkey truth conditions; for example, (4.3a) has a reading on which you end up an expert on *every* famous problem in binding theory x such that you manage to read every article written about x.

4.2.3 The Binder Roof Constraint

Though existential scope-taking is relatively free, there are certain hard constraints. Existential scope of a restricted indefinite over an operator that binds into the indefinite's restrictor is impossible, see (4.4).[1] Following Brasoveanu & Farkas 2011, I'll call this the *Binder Roof Constraint* ('BRC'). (A

1 I'll systematically ignore *functional* readings of indefinites, which can obtain when an indefinite occurs in the scope of an inherently distributive quantifier like *every NP*. In these cases, it seems a sort of wide scope over "functional witnesses" obtains, even when the distributive quantifier binds into the indefinite. See Schlenker 2006 for discussion of these readings in terms of choice functions, Solomon 2011 for criticisms of using choice functions to analyze functional readings, and

number of accounts erroneously predict the existence of such a reading, though the precise truth conditions generated vary depending on the theory. See Section 4.7 for discussion.)

(4.4) No$_i$ candidate submitted a paper he$_i$ had written. (*$\exists > \forall\neg$)
(Schwarz 2001, ex. 25)

Consequences of the BRC can be observed in other domains. Sentence (4.5) lacks a "specific opaque" reading (terminology due to Fodor 1970), i.e. one where the indefinite's restrictor is interpreted relative to Mary's want-worlds (perhaps inexpensive Prada coats don't actually exist, but Mary believes/desires that they do), but where existential quantifier contributed by the indefinite scopes over *wants* (Mary's wanting is *de re*). See e.g. Keshet 2008, who points out that continuing (4.5) with *she saw it last Thursday* requires the thing seen last Thursday to be an actual inexpensive Prada coat (that Mary wants to buy). The impossibility of the specific opaque reading is a consequence of the BRC: *an inexpensive Prada coat* cannot take existential scope over *wants* unless the restrictor is also evaluated above *wants*.[2]

(4.5) Mary wants$_i$ to buy an [inexpensive Prada coat]-w_i. (*$\exists >$ wants)

Notably, the BRC is precisely what an analysis that analyzes existential scope-taking in terms of island-disrespecting QR predicts: QRing *a paper he$_i$ had written* over *no$_i$ candidate* inevitably unbinds the pronoun *he$_i$*. QRing *an inexpensive Prada coat* over *wants* should make it impossible to interpret the restrictor in Mary's want-worlds.

Another consequence of the BRC has not, to my knowledge, been explored in the literature, but provides additional, strong evidence for it. Consider (4.6). It utterly lacks a reading where (i) the existential force of the embedded indefinite *a famous linguist* is within the scope of *not*, and (ii) the existential scope of the object DP as a whole is outside the scope of *not*. (To be sure, it's hard to even imagine what such a reading could amount to. Nevertheless, we will see that other accounts either predict readings where the first *a* takes wide scope and the second one narrow scope or rule such readings out only by stipulation.) In other words, if the object DP takes scope over negation, it must bring *a famous linguist* along for the ride.

(4.6) I didn't read a book by a famous linguist.

Again, this is what using QR for existential scope-taking predicts. Imagine QRing *a famous linguist* out of its container DP to a position below *not* (it doesn't matter for the point if this movement is to a DP adjunction position, as in e.g. Büring 2004, or to a position of clausal scope). Moving the remnant *a book by t_i* over *not* inevitably unbinds the QR trace (since the thing binding the trace was, for the sake of the argument, moved to a position below *not*), and should for this reason be impossible. In contrast, leading analyses of existential scope-taking leave indefinites in situ and create existential scope effects at a distance, thereby making incorrect predictions for cases like (4.6).

Bumford 2013 for an analysis of functional readings and *different* in terms of extensions to the semantics I propose here.
2 But see Szabó 2011 for an opposing view.

4.3 Evaluation and scope islands

4.3.1 Evaluation

As we explored in Chapter 3, continuized grammars evaluate a derived scopal meaning by feeding to it a *trivial* scope or continuation. Along these lines, let us define a notion that will help us talk about when a constituent has been evaluated. See Definition 4.1. The basic idea is simply that an evaluated type is the type of a denotation that has taken as much scope as it possibly can—in other words, the type of a denotation whose continuation arguments have all been saturated.

Definition 4.1 (Evaluated types). The set of evaluated types E is defined as follows: for any α, β: $\mathsf{K}_{\mathsf{M}\beta}\ \alpha \notin E$, and for any type α, if α mentions a non-evaluated type, $\alpha \notin E$. Every other type is in E.[3]

Derivatively, a *constituent* is considered evaluated iff that constituent denotes something with an evaluated type.

4.3.2 Scope islands

I adopt the standard assumption that scope-taking in natural language is delimited by scope islands. The usual way this is implemented in LF-based frameworks is via a representational constraint starring LFs where something has QR'd out of a tensed clause. Continuized grammars have in situ scope-taking and no use for LF, but they have an alternative for bounding scope-taking. My proposal for scope islands can be summed up as follows: a scope island is any constituent that *must be evaluated*—that is, a constituent with an evaluated type, one whose continuation arguments have all been saturated:

Definition 4.2 (Scope islands). *Scope islands* are constituents that must be evaluated.

A corollary of Definition 4.2—which bears an intriguing resemblance to Chomsky's 2008 conception of *phases* as the computational domains of natural language[4]—is that scope island are obligatory Lowered. The requirement that scope islands must be evaluated is intended as nothing more and nothing less than the denotational correlate of prohibiting QR out of scope islands.

4.3.3 Examples

Because evaluated meanings are often part of larger derivations which presuppose scopal combination, obligatory evaluation at a scope island will often be paired with re-Lifting in the form of a Reset. Importantly, Resetting a scopal program whose scopal tier consists solely of monadic sequencing *has no effect*. The input to and output of Reset are exactly the same:

3 This definition is refined and made more explicit in Appendix A, but it will suffice for present purposes.

4 Of course, aside from tensed clauses, Chomsky takes *v*P and CP to be phases as well. I will leave it open how closely the theory I'm proposing maps onto Chomsky's notion of a phase.

Fact 4.1 (Reset and monadic programs).

$$\left(\frac{m_v \multimap [\,]}{f\,v}\right)^{\downarrow\uparrow} = \frac{m_v \multimap [\,]}{f\,v}$$

Proof.

$$\begin{aligned}
\left(\frac{m_v \multimap [\,]}{f\,v}\right)^{\downarrow\uparrow} &= (m_v \multimap (f\,v)^{\eta})^{\uparrow} && \downarrow \\
&= \frac{(m_v \multimap (f\,v)^{\eta})_u \multimap [\,]}{u} && \uparrow \\
&= \frac{m_v \multimap (f\,v)^{\eta}_u \multimap [\,]}{u} && \text{Assoc} \\
&= \frac{m_v \multimap [\,]}{f\,v} && \text{LeftID}
\end{aligned}$$

□

More generally, because ↑ is identified with ⊸, any side effects that survive evaluation are free to take scope post-evaluation: sequencing an evaluated, impure program π with a new, post-evaluation continuation/scope k means that any side effects in π inevitably influence the evaluation of k. Thus, for example, Reset has *no effect* on e.g. *a linguist left*, see (4.2). Evaluation brings us to a fully evaluated expression, and Lifting brings us back into a scopal expression. Nothing, however, has changed. The indefiniteness retains scope over its continuation.

Fact 4.2 (Resetting *a linguist left*).

$$\left(\frac{\textbf{a.ling}_x \multimap [\,]}{\mathsf{left}\,x}\right)^{\downarrow\uparrow} = \frac{\textbf{a.ling}_x \multimap [\,]}{\mathsf{left}\,x}$$

Similarly for *she left*. Reset has no effect:

Fact 4.3 (Resetting *she left*).

$$\left(\frac{\textbf{pro}_x \multimap [\,]}{\mathsf{left}\,x}\right)^{\downarrow\uparrow} = \frac{\textbf{pro}_x \multimap [\,]}{\mathsf{left}\,x}$$

However, crucially, the situation is quite different for *every linguist left*, see Fact 4.4. Evaluating the derived meaning for this sentence yields a trivial $\mathsf{M}t$ program which just bottles up the truth condition that every linguist left and lacks either nondeterministic or state-changing effects. Evaluation thus utterly discharges the scope-taking ability of the universal. Even after Lifting back into a tower, the new continuation scopes over the universal: the quantificational force that inhered in the pre-evaluated meaning is relegated to a truth condition on the bottom of the derived tower, and that is where its role ends.

Fact 4.4 (Resetting *every linguist left*).

$$\begin{aligned}
\left(\frac{\textbf{every.ling}\,(\lambda x.\,[\,])}{\mathsf{left}\,x}\right)^{\downarrow\uparrow} &= (\textbf{every.ling}\,(\lambda x.\,(\mathsf{left}\,x)^{\eta}))^{\uparrow} && \downarrow \\
&= ((\forall x.\,\mathsf{ling}\,x \Rightarrow \mathsf{left}\,x)^{\eta})^{\uparrow} && \textbf{ev.ling} \\
&= \frac{(\forall x.\,\mathsf{ling}\,x \Rightarrow \mathsf{left}\,x)^{\eta}_{p} \multimap [\,]}{p} && \uparrow \\
&= \frac{[\,]}{\forall x.\,\mathsf{ling}\,x \Rightarrow \mathsf{left}\,x} && \text{LeftID}
\end{aligned}$$

I give some additional examples to underline the intuition. First, in Fact 4.5 I apply Reset to a tower very much like the one we derived for the surface-scope reading of *a man met every linguist* in (3.24). (The input to Reset here differs slightly from what we derived in (3.24) in that I've applied the Bind type-shifter to the subject.) Evaluating the two-level tower (the first step in the Reset) gives a dynamic proposition which retains the nondeterministic and state-changing side effects associated with the indefinite, even as the universal quantifier is discharged. Lifting (completing the Reset) brings the nondeterministic and state-changing side effects incurred by the indefinite back onto the scopal tier, while leaving the universally quantified value on the bottom level.

Fact 4.5 (Resetting *a man met every linguist*).

$$\begin{aligned}
\left(\frac{\textbf{a.man}^{\triangleright}_{x} \multimap \textbf{ev.ling}\,(\lambda y.[\,])}{\mathsf{met}\,y\,x}\right)^{\downarrow\uparrow} &= (\textbf{a.man}^{\triangleright}_{x} \multimap \textbf{ev.ling}\,(\lambda y.\,(\mathsf{met}\,y\,x)^{\eta}))^{\uparrow} && \downarrow \\
&= (\textbf{a.man}^{\triangleright}_{x} \multimap (\forall y.\,\mathsf{ling}\,y \Rightarrow \mathsf{met}\,y\,x)^{\eta})^{\uparrow} && \textbf{ev.ling} \\
&= \frac{(\textbf{a.man}^{\triangleright}_{x} \multimap (\forall y.\,\mathsf{ling}\,y \Rightarrow \mathsf{met}\,y\,x)^{\eta})_{p} \multimap [\,]}{p} && \uparrow \\
&= \frac{\textbf{a.man}^{\triangleright}_{x} \multimap (\forall y.\,\mathsf{ling}\,y \Rightarrow \mathsf{met}\,y\,x)^{\eta}_{p} \multimap [\,]}{p} && \text{Assoc} \\
&= \frac{\textbf{a.man}^{\triangleright}_{x} \multimap [\,]}{\forall y.\,\mathsf{ling}\,y \Rightarrow \mathsf{met}\,y\,x} && \text{LeftID}
\end{aligned}$$

Let's go through another example, in Fact 4.6. This time, we apply Reset to the *inverse* scope derivation of *a man met every linguist*, specifically to the two-story tower we derived after the first application of Lower in example (3.28). Since *every linguist* takes widest scope, the evaluated meaning to be re-Lifted is deterministic and anaphorically inert; **ev.ling** uses the nondeterminism and stack modification in the object indefinite in the service of calculating a truth condition and then simply returns a pure computation whose value is that truth condition. Since the result of evaluation is a pure program, re-Lifting cannot reanimate the nondeterministic or state-changing side effects incurred by the indefinite. Because those side effects do not survive evaluation, they cannot take scope after evaluation.

Fact 4.6 (Resetting *a man met every linguist* on the inverse scope reading).

$$\begin{aligned}
\left(\frac{\dfrac{\textbf{ev.ling}\,(\lambda y.[\,])}{\textbf{a.man}^{\triangleright}_{x} \multimap [\,]}}{\mathsf{met}\;y\;x}\right)^{\Downarrow\uparrow} &= \textbf{ev.ling}\,(\lambda y.\,\textbf{a.man}^{\triangleright}_{x} \multimap (\mathsf{met}\;y\;x)^{\eta}) && \Downarrow \\
&= (\forall y.\,\mathsf{ling}\;y \Rightarrow \exists x.\,\mathsf{man}\;x \wedge \mathsf{met}\;y\;x)^{\eta} && \textbf{ev.ling} \\
&= \frac{(\forall y.\,\mathsf{ling}\;y \Rightarrow \exists x.\,\mathsf{man}\;x \wedge \mathsf{met}\;y\;x)^{\eta}_{p} \multimap [\,]}{p} && \uparrow \\
&= \frac{[\,]}{\forall y.\,\mathsf{ling}\;y \Rightarrow \exists x.\,\mathsf{man}\;x \wedge \mathsf{met}\;y\;x} && \text{LeftID}
\end{aligned}$$

As a final point, notice that while **every.ling** is dynamically closed with respect to any drefs or nondeterminism introduced in its scope, it does not eliminate the side effects incurred by unbound pronouns (that is, both before and after evaluation, the pronoun remains unbound). See Fact 4.7, where we Reset a two-level tower derived for the sentence *every linguist met her*. While the Reset discharges the universal, the stack-sensitivity that inheres in the semantics of **pro** lives to fight another day. Because the result of evaluating *every linguist met her* still depends on the stack to fix the value of **pro**, the effect is as if **pro** has acquired scope over the entire program returned on evaluation. Put another way: the effect of evaluating **every.ling** is to wipe out all the nondeterminism and drefs in its scope, but not to wipe out any anaphoric sensitivity.

Fact 4.7 (Resetting *every linguist met her*).

$$\begin{aligned}
\left(\frac{\textbf{ev.ling}\,(\lambda x.\,\textbf{pro}_{y} \multimap [\,])}{\mathsf{met}\;y\;x}\right)^{\downarrow\uparrow} &= (\textbf{ev.ling}\,(\lambda x.\,\textbf{pro}_{y} \multimap (\mathsf{met}\;y\;x)^{\eta}))^{\uparrow} && \downarrow \\
&= (\textbf{pro}_{y} \multimap (\forall x.\,\mathsf{ling}\;x \Rightarrow \mathsf{met}\;y\;x)^{\eta})^{\uparrow} && \textbf{ev.ling} \\
&= \frac{(\textbf{pro}_{y} \multimap (\forall x.\,\mathsf{ling}\;x \Rightarrow \mathsf{met}\;y\;x)^{\eta})_{p} \multimap [\,]}{p} && \uparrow \\
&= \frac{\textbf{pro}_{y} \multimap (\forall x.\,\mathsf{ling}\;x \Rightarrow \mathsf{met}\;y\;x)^{\eta}_{p} \multimap [\,]}{p} && \text{Assoc} \\
&= \frac{\textbf{pro}_{y} \multimap [\,]}{\forall x.\,\mathsf{ling}\;x \Rightarrow \mathsf{met}\;y\;x} && \text{LeftID}
\end{aligned}$$

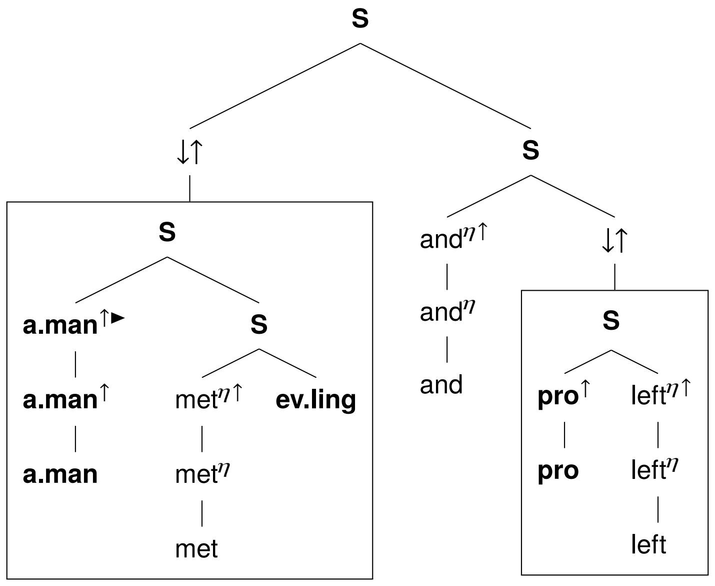
4.4 Taking scope after evaluation

As the previous section makes clear, though the grammar insists scope islands be evaluated (i.e., in representational terms, that nothing QRs outside of the scope island), in many cases side effects *survive evaluation* to take scope beyond the domains of the scope island. The upshot is a *semantic explanation* for why indefinites can impose both anaphoric and nondeterministic effects on constituents outside of a scope island. Thus, we predict that indefinites can both nondeterministically bind into subsequent sentences and take nondeterministic scope over structurally higher operators outside their nearest island. In addition, because the way side effects percolate out of islands is fundamentally scopal, we predict that exceptional scope feeds binding and derive the BRC.

4.4.1 Cross-sentential binding

I'll first show how we can derive cross-sentential anaphora in a simple case like *a_i man met every linguist, and (then) he_i left*. The derivation, including a full analysis tree, is given below. Since both *a man met every linguist* and *he left* are scope islands (this is the significance of the boxed nodes in the analysis tree), each has to be Reset. The Reset of the surface-scope derivation of *a man met every linguist* (with a Bind-shifted indefinite) was derived in Fact 4.5, and the Reset of a sentence like *he left* was derived in Fact 4.3. After Lifting *and*, combination happens via two instances of scopal functional application. This brings **pro** within the immediate scope of the dref-hosting program $\textbf{a.man}^{\triangleright}$, and binding is achieved. (As in Chapter 2, the theory derives the correct interpretation without availing itself of a lexical dynamic conjunction.) We conclude by Resetting the derived conjunction in order to illustrate that the indefiniteness survives Reset again; indeed, an indefinite will *always* survive Reset so long as it never gets caught within the scope of an operator that discharges the indefinite's effects on evaluation.

(4.7)



$$\left(\begin{array}{c} \frac{\mathsf{M}t}{t} \\ \\ \text{a man met every ling} \\ \\ \frac{\textbf{a.man}^{\triangleright}_x \multimap [\,]}{\forall y.\, \mathsf{ling}\, y \Rightarrow \mathsf{met}\, y\, x} \end{array} \left(\begin{array}{cc} \frac{\mathsf{M}t}{t \to t \to t} & \frac{\mathsf{M}t}{t} \\ \\ \text{and} & \text{he left} \\ \\ \frac{[\,]}{\mathsf{and}} & \frac{\textbf{pro}_z \multimap [\,]}{\mathsf{left}\, z} \end{array}\right)\right) = \begin{array}{c} \frac{\mathsf{M}t}{t} \\ \\ \text{a man met every ling, and he left} \\ \\ \frac{\textbf{a.man}^{\triangleright}_x \multimap \textbf{pro}_z \multimap [\,]}{\forall y.\, \mathsf{ling}\, y \Rightarrow \mathsf{met}\, y\, x \wedge \mathsf{left}\, z} \end{array}$$

$$\begin{array}{lcl} \rightsquigarrow & \textbf{a.man}^{\triangleright}_x \multimap \textbf{pro}_z \multimap (\forall y.\, \mathsf{ling}\, y \Rightarrow \mathsf{met}\, y\, x \wedge \mathsf{left}\, z)^{\eta} & \downarrow \\ = & \textbf{a.man}^{\triangleright}_x \multimap (\forall y.\, \mathsf{ling}\, y \Rightarrow \mathsf{met}\, y\, x \wedge \mathsf{left}\, x)^{\eta} & \text{Binding} \\ \rightsquigarrow & \frac{\textbf{a.man}^{\triangleright}_x \multimap [\,]}{\forall y.\, \mathsf{ling}\, y \Rightarrow \mathsf{met}\, y\, x \wedge \mathsf{left}\, x} & \uparrow \end{array}$$

It isn't possible for a Bind-shifted universal DP such as *every linguist* to bind out of a sentence in an analogous way. Because *every linguist* is dynamically closed, any anaphoric potential that inheres in it pre-evaluation disappears on evaluation. Moreover, cross-sentential anaphora to *a man* is correctly precluded when *every linguist* takes inverse scope over it. As we saw in Fact 4.6, the binding potential and nondeterminism that inhered in the indefinite do not in that case survive evaluation.

4.4.2 Nondeterministic exceptional scope over higher operators

Post-Reset scope-taking also allows side effects to scope out of islands and over structurally higher operators outside the island. An example with an indefinite's nondeterminism taking exceptional scope over a negation—e.g. in order to derive an $\exists > \neg$ reading for *John said it wasn't the case that a relative of mine had met every linguist*—is given in (4.8). In addition to Resetting the embedded tensed clause, I've also applied Lift to *it wasn't the case that*, which I'll assume has the same semantics as negation. Combination happens via one instance of scopal functional application (this also requires us to apply η inside the Reset tower, since **not** is expecting a Mt as its argument). We conclude the derivation by applying Lift to the bottom story and executing a one-fell-swoop Lower.

(4.8)
$$\left(\begin{array}{cc} \frac{\mathsf{M}t}{\mathsf{M}t \to \mathsf{M}t} & \frac{\mathsf{M}t}{\mathsf{M}t} \\ \\ \text{NEG} & \text{a rel met every linguist} \\ \\ \frac{[\,]}{\textbf{not}} & \frac{\textbf{a.rel}_x \multimap [\,]}{(\forall y.\, \mathsf{ling}\, y \Rightarrow \mathsf{met}\, y\, x)^{\eta}} \end{array}\right) = \begin{array}{c} \frac{\mathsf{M}t}{\mathsf{M}t} \\ \\ \text{NEG a rel met every linguist} \\ \\ \frac{\textbf{a.rel}_x \multimap [\,]}{\textbf{not}\, (\forall y.\, \mathsf{ling}\, y \Rightarrow \mathsf{met}\, y\, x)^{\eta}} \end{array}$$

$$\begin{array}{lcl} \rightsquigarrow & \textbf{a.rel}_x \multimap \textbf{not}\, (\forall y.\, \mathsf{ling}\, y \Rightarrow \mathsf{met}\, y\, x)^{\eta} & \uparrow \text{ applied to lower level, } \downdownarrows \\ = & \textbf{a.rel}_x \multimap (\neg\forall y.\, \mathsf{ling}\, y \Rightarrow \mathsf{met}\, y\, x)^{\eta} & \textbf{not} \end{array}$$

This derivation gives the embedded indefinite's nondeterminism scope over negation, even though a scope island intervenes between the indefinite's base position and the negation. The result is that we flip the polarity of $\forall y.\, \mathsf{ling}\, y \Rightarrow \mathsf{met}\, y\, x$, for every relative of mine x.

We can also give an indefinite's nondeterminism exceptional scope out of the antecedent of a conditional. See (4.9) for an analysis of (4.1a). We've Reset the antecedent and applied Lift to *if* and *I'm rich* (which has also been Reset, as it is a tensed clause). Because *if* is expecting dynamic propositions as arguments, we apply η to the bottom stories of the antecedent and consequent towers. After combination and evaluation (see Appendix A for the full details), we derive a program where the indefinite's nondeterminism scopes over the conditional. Despite being located inside a scope island, the indefinite is able to acquire scope over the conditional.

(4.9)
$$\left(\begin{array}{c}\dfrac{\mathsf{M}t}{\mathsf{M}t \to \mathsf{M}t \to \mathsf{M}t}\\[2ex] \text{if}\\[2ex] \dfrac{[\,]}{\textbf{if}}\end{array}\left(\begin{array}{cc}\dfrac{\mathsf{M}t}{\mathsf{M}t} & \dfrac{\mathsf{M}t}{\mathsf{M}t}\\[2ex] \text{a relative dies} & \text{I'm rich}\\[2ex] \dfrac{\textbf{a.rel}_x \multimap [\,]}{(\mathsf{dies}\,x)^{\eta}} & \dfrac{[\,]}{(\mathsf{rich\,me})^{\eta}}\end{array}\right)\right) = \begin{array}{c}\dfrac{\mathsf{M}t}{\mathsf{M}t}\\[2ex] \text{if a rel dies I'm rich}\\[2ex] \dfrac{\textbf{a.rel}_x \multimap [\,]}{\textbf{if}\,(\mathsf{dies}\,x)^{\eta}\,(\mathsf{rich\,me})^{\eta}}\end{array}$$

$$\begin{aligned} &\rightsquigarrow \quad \textbf{a.rel}_x \multimap \textbf{if}\,(\mathsf{dies}\,x)^{\eta}\,(\mathsf{rich\,me})^{\eta} \quad \uparrow, \Downarrow \\ &= \quad \textbf{a.rel}_x \multimap (\mathsf{dies}\,x \Rightarrow \mathsf{rich\,me})^{\eta} \quad \textbf{if} \end{aligned}$$

A note on the truth conditions. The derived $\mathsf{M}t$ program returns a True boolean iff $\exists x.\, \mathsf{rel}\, x \wedge (\mathsf{dies}\, x \Rightarrow \mathsf{rich\,me})$; in English, there is a relative of mine x such that if x dies, I'm rich. In fact the truth condition is a little too weak: we only need to find one of my relatives who doesn't die in order to satisfy it (change *dies* to *dies tomorrow* to see why this is bad). However, this weakness is easily avoided if the conditional operator has a slightly more sophisticated semantics that does not allow for vacuous satisfaction (Szabolcsi 2010: 95; as Szabolcsi notes, this treatment should be extended to universal DPs, which presuppose the non-emptiness of their restrictor and thus cannot be vacuously satisfied).[5] Modulo the inadequate treatment of the conditional, the analysis illustrates how the correct prediction is made. Reset lets the indefinite's nondeterminism acquire post-evaluation scope over the conditional operator (but will restrict the scope of a universal to its minimal tensed clause).

A point by way of wrapping up. Once we've Reset a meaning, we're of course free to keep fiddling with it. For instance, we might internally Lift the output of Fact 4.5, as in (4.10). The effect is to move the indefinite's effects to the top tier of a three-story tower, as if the indefinite itself is getting ready to take inverse scope. Despite the application of internal Lift, the universal remains trapped in a static truth condition on the bottom floor. Thus, post-Reset, any side effects which survive evaluation are free to take inverse scope via the same mechanisms which allow them to take inverse scope inside their minimal tensed clause. This will allow us to readily account for data like the $\exists > \forall$ reading of *every man claims that if a relative of mine dies, I'll inherit a fortune*. On the other hand, if the

5 Note that while these moves vitiate some of Reinhart 1997's objections against existential closure at a distance (see e.g. Heim 1982, Szabolcsi 2003), some remain valid, namely cases like *the class did not understand some argument*.

Reset tower *isn't* internally Lifted, the indefinite's side effects will end up out-scoped by any scopal expressions to the left. This will allow us to derive cases of intermediate exceptional scope, as in examples such as (4.2a).

(4.10)
$$\left(\begin{array}{c} \dfrac{\mathsf{M}t}{t} \\[1em] \text{a man met every linguist} \\[1em] \dfrac{\textbf{a.man}^{\triangleright}_x \multimap [\,]}{\forall y.\, \mathsf{ling}\, y \Rightarrow \mathsf{met}\, y\, x} \end{array}\right)^{\upuparrows} = \begin{array}{c} \dfrac{\mathsf{M}t}{\dfrac{\mathsf{M}t}{t}} \\[1em] \text{a man met every linguist} \\[1em] \dfrac{\textbf{a.man}^{\triangleright}_x \multimap [\,]}{\dfrac{[\,]}{\forall y.\, \mathsf{ling}\, y \Rightarrow \mathsf{met}\, y\, x}} \end{array}$$

4.4.3 Feeding binding and deriving the BRC

We now have a sense of the basic contours of the proposal. Exceptional scope-taking happens when side effects take scope after evaluation. Semantically, the way exceptionally scoping side effects interact with their context post-evaluation is no different from the sort of scope-taking that characterizes, for instance, inverse scope within a tensed clause.

This makes a number of good predictions. For one, it follows that exceptional scope-taking over dynamically closed operators feeds anaphora: when an indefinite's side effects take exceptional scope over an operator like *not* or *if*, as in the previous section, this inevitably brings any discourse referents the indefinite may have induced along for the ride. Consider a case like *if a relative of mine dies, I'm rich; she's a steel magnate*. Binding is only possible when the indefinite's side effects take exceptional scope out of the antecedent of the conditional: exceptional scope feeds anaphora. Our analysis goes much as in the case of cross-sentential anaphora we derived earlier in (4.7). We Reset each of the conjuncts and combine everything up with a Lifted static conjunction. Exceptional scope-taking of the indefinite's side effects includes the indefinite's dref, the pronoun ends up within the immediate scope of the dref-hosting program $\textbf{a.rel}^{\triangleright}$, and binding is achieved. We thus derive that scope-taking feeds anaphora.

(4.11)
$$\left(\begin{array}{c} \dfrac{\mathsf{M}t}{t} \\[1em] \text{if a relative dies I'm rich} \\[1em] \dfrac{\textbf{a.rel}^{\triangleright}_x \multimap [\,]}{\mathsf{dies}\, x \Rightarrow \mathsf{rich}\, \mathsf{me}} \end{array} \left(\begin{array}{cc} \dfrac{\mathsf{M}t}{t \to t \to t} & \dfrac{\mathsf{M}t}{t} \\[1em] ; & \text{she's a steel magnate} \\[1em] \dfrac{[\,]}{\mathsf{and}} & \dfrac{\textbf{pro}_y \multimap [\,]}{\mathsf{steel.mag}\, y} \end{array}\right)\right)$$

$$\rightsquigarrow \quad \textbf{a.rel}^{\triangleright}_x \multimap ((\mathsf{dies}\, x \Rightarrow \mathsf{rich}\, \mathsf{me}) \wedge \mathsf{steel.mag}\, x)^{\eta} \quad \textbf{S}, \downarrow, \text{Binding}$$

Similarly, because the scope-taking of side effects post-evaluation is modeled via bona fide scope mechanisms, we automatically derive the BRC (i.e. that the indefinite's side effects cannot scope over an operator that binds into the indefinite's restrictor). Consider the attempted derivation of Schwarz's 2001 *no candidate*$_i$ *submitted a paper he*$_i$ *wrote* in (4.12). We have Bind-shifted *no candidate* and internally Lifted the indefinite DP *a paper he wrote* (whose derivation I gloss over but whose semantics amounts to the M*e* program $\textbf{pro}_z \multimap \textbf{a.paper.by}\ z$) in an attempt to give it scope over an operator that binds into its restrictor. Two applications of higher-order scopal functional application manage to glue the pieces together, and the nondeterminism of the indefinite does indeed out-scope *no candidate*, but this forces **pro** to be evaluated outside the scope of *no candidate*. While the result certainly represents a possible reading of *no candidate submitted a paper he wrote*, it is not the bound-pronoun, wide-scope indefinite reading.

(4.12)
$$\left(\begin{array}{c} \dfrac{\mathsf{M}t}{\dfrac{\mathsf{M}t}{e}} \\[2ex] \text{no candidate} \\[2ex] \dfrac{[\,]}{\dfrac{\textbf{no.cand}\,(\lambda x.\, x_v^{\eta\triangleright} \multimap [\,])}{v}} \end{array} \left(\begin{array}{c} \dfrac{\mathsf{M}t}{\dfrac{\mathsf{M}t}{e \to e \to t}} \\[2ex] \text{submitted} \\[2ex] \dfrac{[\,]}{\dfrac{[\,]}{\mathsf{submitted}}} \end{array} \quad \begin{array}{c} \dfrac{\mathsf{M}t}{\dfrac{\mathsf{M}t}{e}} \\[2ex] \text{a paper he wrote} \\[2ex] \dfrac{\textbf{pro}_z \multimap (\textbf{a.paper.by}\ z)_y \multimap [\,]}{\dfrac{[\,]}{y}} \end{array}\right)\right)$$

$$\begin{array}{rll} \rightsquigarrow & \textbf{pro}_z \multimap (\textbf{a.paper.by}\ z)_y \multimap \textbf{no.cand}\,(\lambda x.\, x_v^{\eta\triangleright} \multimap (\mathsf{submitted}\ y\ v)^{\eta}) & \mathbb{S}, \downdownarrows \\ = & \textbf{pro}_z \multimap (\textbf{a.paper.by}\ z)_y \multimap (\neg\exists x.\, \mathsf{cand}\ x \wedge \mathsf{submitted}\ y\ x)^{\eta} & \textbf{no.cand} \end{array}$$

Similarly, we do not over-generate readings for *I didn't read a book by a famous linguist*. Recall that we want to avoid making the prediction that the object DP as a whole can take wider existential scope than the embedded indefinite *a famous linguist*. Our semantics allows two scope orderings for *a book by a famous linguist*, one where *a famous linguist* and the object DP as a whole take scope together, and one where the two scope separately, but in which *a famous linguist* out-scopes the object DP as a whole.

$$\begin{array}{c} \dfrac{\mathsf{M}t}{e} \\[2ex] \text{a book by a famous linguist} \\[2ex] \dfrac{\textbf{a.fam.ling}_y \multimap (\textbf{a.book.by}\ y)_x \multimap [\,]}{x} \end{array} \qquad \begin{array}{c} \dfrac{\mathsf{M}t}{\dfrac{\mathsf{M}t}{e}} \\[2ex] \text{a book by a famous linguist} \\[2ex] \dfrac{\textbf{a.fam.ling}_y \multimap [\,]}{\dfrac{(\textbf{a.book.by}\ y)_x \multimap [\,]}{x}} \end{array}$$

On either of these derivations, it's impossible for the the object DP as a whole to out-scope an operator that out-scopes *a famous linguist*. In the first, the two indefinites necessarily scope together. In the second, there is a possibility of an operator intervening between *a famous linguist* and the "remnant" indefinite, but only with the former taking widest scope; any attempt to scope the "remnant" over an operator will inevitably bring *a famous linguist* along for the ride over that operator. As with QR, there is no possibility of scoping the embedded indefinite outside the container DP, and then scoping the "remnant" over it.[6]

4.5 Extensions

4.5.1 Plural indefinites

Like singular indefinites, plural indefinites take exceptional existential scope. Sentence (4.13a) has a reading entailing that I have two relatives such that if they each die, I'll inherit a house, and sentence (4.13b) can mean that there's a plurality of relatives of mine such that if they each die, I'll inherit a house. However, (4.13a) and (4.13b) lack readings in which the two/some relatives are *each* such that them dying nets me a house (Ruys 1992): though existential scope-taking is unbounded, a plural indefinite cannot take distributive scope out of its tensed clause (here, the antecedents of the conditionals; '$\delta >$ if' signifies an interpretation where distribution over the atoms in the relevant plurality scopes above the conditional's antecedent).

(4.13) a. If ⟨two relatives of mine die⟩, I'll inherit a house. ($\exists >$ if; $*\delta >$ if)
       b. If ⟨some relatives of mine die⟩, I'll inherit a house. ($\exists >$ if; $*\delta >$ if)

As with singular indefinites, exceptional existential scope-taking by plural indefinites feeds anaphora. Sentences (4.13a) and (4.13b) can be continued with *their lawyer told me*, and (4.14) is licit on the indicated indexing when the indefinite's side effects take either intermediate or maximal exceptional existential scope (note that the relevant exceptional scope readings are still not distributive; these readings inevitably talk about co-authored books), although (4.14) is infelicitous when *two famous linguists* scopes within its minimal relative clause.

(4.14) Every student who read every book written by two$_i$ famous linguists praised them$_i$.

We can derive these facts with some basic meanings for plural indefinites and distributivity operators. For concreteness, I'll model plural individuals as sets of atomic individuals. Thus, the domain of plural individuals is the powerset of the domain of atomic individuals minus the empty set. See Definition 4.3. Pluralized inherently distributive predicates distribute over the atomic individuals in their argument—i.e. $\textsf{rels.of.me} \coloneqq \lambda X.\, \forall x.\, x \in X \Rightarrow \textsf{rel.of.me}\, x$ (Link 1998, Krifka 1989 and many others; I give a definition for a distributivity operator that can be used to build plural predicates in (4.5))—while singular predicates only have atomic individuals in their domain (following Schwarzschild 1996, I identify $\{x\}$ and x and assume that predicates by default are n-ary relations over plural individuals).

6 In fact, this treatment modestly improves on QR in that, since scope-taking is done in situ, there isn't even the *possibility* of unbinding a trace via an illicit remnant scoping—something which QR-based theories must rule out via a representational prohibition on LFs with unbound traces and/or a ban on vacuous quantification.

Definition 4.3 (Domain of plural individuals).

$$\mathcal{D}_e^{\mathsf{+pl}} := 2^{\mathcal{D}_e} \backslash \{\emptyset\}$$

Given these pieces, the semantics of the plural indefinites *two buildings* and *some relatives of mine* are as in Definition 4.4. These meanings are just the plural counterparts of the meanings for singular indefinites like *a linguist*. Like singular indefinites, both of these meanings are nondeterministic programs in the State.Set monad. Thus, as with singular indefinites, this nondeterminism will persist post-evaluation and is free to scope out of scope islands; having nondeterministic meanings for plural indefinites immediately predicts that plural indefinites take exceptional existential scope in precisely the same way as singular indefinites.

Definition 4.4 (Plural indefinites).

$$\begin{array}{cc} \mathsf{M}e & \mathsf{M}e \\ \text{two buildings} & \text{some relatives of mine} \\ \lambda s.\,\{\langle X,\ s\rangle \mid \mathsf{bldgs}\,X \wedge |X| = 2\} & \lambda s.\,\{\langle X,\ s\rangle \mid \mathsf{rels.of.me}\,X \wedge |X| \geq 1\} \end{array}$$

As these entries make clear, I don't assume plural indefinites are inherently distributive (pace Reinhart 1997). It follows that wide existential scope of a plural indefinite's nondeterministic side effects does not entail that the plural indefinite takes wide distributive scope. (Making plural indefinites inherently distributive would make it difficult to explain how we can construe *two/some relatives of mine lifted a piano together* collectively. See e.g. Kamp & Reyle 1993.) Instead, distributivity must come in via other means. As is standard, we'll assume phonologically null distributivity operators in the syntax. My entry for the silent distributivity operator δ, in Definition 4.5, combines with a predicate P and a plural individual X to yield a tower that universally quantifies over the atoms in X. Note that this entry is totally polymorphic; there is thus no need for generalized distributivity operators à la Lasersohn 1998.[7]

Definition 4.5 (Distributivity operator).

$$(e \to \alpha) \to e \to \frac{\mathsf{M}t}{\alpha}$$

$$\delta$$

$$\mathbf{D} := \lambda RX.\, \frac{\lambda s.\,\{\langle \forall x.\, x \in X \Rightarrow \exists s'.\, \langle \mathsf{True},\ s'\rangle \in [\ ]\, s,\ s\rangle\}}{R\,x}$$

The reason to give distributivity operators a scopal semantics in the first place is that plural indefinites (and plurals more generally) can take distributive inverse scope. Example (4.15) (taken

7 Generally, α will be some type ending in t, though that actually isn't forced by the semantics; any function over individuals will do. This may be useful for the distributive interpretation of *two linguists' mothers met*.

from Reinhart 2006: 77) can mean that two buildings are guarded, i.e. has a reading on which guards vary with buildings.

(4.15) A guard is standing in front of two buildings. $(2 > \delta > \exists)$

Deriving this reading is straightforward. We begin by assembling the VP. The distributivity operator δ combines with *is standing in front of* (which I treat as an unanalyzed unit *fronts*), then with *two buildings* via Lifting and scopal functional application:

(4.16)

$$\left(\begin{array}{cc} \dfrac{\mathsf{M}t}{e \to \dfrac{\mathsf{M}t}{e \to t}} & \dfrac{\mathsf{M}t}{e} \\[2ex] \delta \text{ fronts} & \text{two buildings} \\[2ex] \dfrac{[\,]}{\lambda Y.\, \dfrac{\mathbf{D}\,\mathsf{fronts}\, Y\, (\lambda P.\, [\,])}{P}} & \dfrac{\mathbf{2.bldgs}_Y \multimap [\,]}{Y} \end{array}\right) = \begin{array}{c} \dfrac{\mathsf{M}t}{\dfrac{\mathsf{M}t}{e \to t}} \\[2ex] \delta \text{ fronts two buildings} \\[2ex] \dfrac{\mathbf{2.bldgs}_Y \multimap [\,]}{\dfrac{\mathbf{D}\,\mathsf{fronts}\, Y\, (\lambda P.\, [\,])}{P}} \end{array}$$

Next, let's see what happens when we combine up the whole sentence and evaluate. We apply internal Lift so that the distributive quantifier $\mathbf{D}$ can take inverse scope over the subject. After externally Lifting the subject twice, combination is via super-higher-order scopal functional application, which works as expected (and follows from the grammar as defined already). We Lower in one fell swoop by applying the 4-level tower to $\Downarrow\!\Downarrow\!\Downarrow\; = \lambda M.\, M\, (\lambda m.\, m\, \nu)$:[8]

(4.17)

$$\left(\begin{array}{cc} \dfrac{\mathsf{M}t}{\dfrac{\mathsf{M}t}{\dfrac{\mathsf{M}t}{e}}} & \dfrac{\mathsf{M}t}{\dfrac{\mathsf{M}t}{\dfrac{\mathsf{M}t}{e \to t}}} \\[2ex] \text{a guard} & \delta \text{ fronts two buildings} \\[2ex] \dfrac{[\,]}{\dfrac{[\,]}{\dfrac{\mathbf{a.guard}_x \multimap [\,]}{x}}} & \dfrac{\mathbf{2.bldgs}_Y \multimap [\,]}{\dfrac{\mathbf{D}\,\mathsf{fronts}\, Y\, (\lambda P.\, [\,])}{\dfrac{[\,]}{P}}} \end{array}\right) = \begin{array}{c} \dfrac{\mathsf{M}t}{\dfrac{\mathsf{M}t}{\dfrac{\mathsf{M}t}{t}}} \\[2ex] \delta \text{ fronts two buildings} \\[2ex] \dfrac{\mathbf{2.bldgs}_Y \multimap [\,]}{\dfrac{\mathbf{D}\,\mathsf{fronts}\, Y\, (\lambda P.\, [\,])}{\dfrac{\mathbf{a.guard}_x \multimap [\,]}{P\, x}}} \end{array}$$

$$\begin{aligned} &\rightsquigarrow \quad \mathbf{2.bldgs}_Y \multimap \mathbf{D}\,\mathsf{fronts}\, Y\, (\lambda P.\, \mathbf{a.guard}_x \multimap (P\, x)^{\eta}) \quad \Downarrow\!\Downarrow\!\Downarrow \\ &= \quad \mathbf{2.bldgs}_Y \multimap (\forall y.\, y \in Y \Rightarrow \exists x.\, \mathsf{guard}\, x \wedge \mathsf{fronts}\, y\, x)^{\eta} \quad \mathbf{D} \end{aligned}$$

The plural indefinite's nondeterministic side effects survive evaluation, while the dynamically closed

8 Inverse-scope distributive readings of plurals are thought to be more restricted than allowing **D** to be freely available would predict. See e.g. Reinhart 2006: 113–123 and Szabolcsi 2010: 115–116 for discussion.

distributivity operator and everything in its scope are relegated to a static truth condition (i.e., given a plurality of two buildings Y, for each atomic building y in Y, there's a guard standing in front of y). Like a universal DP, the scope-taking ability of δ is discharged on evaluation/Reset. Distributivity is thus correctly bounded by scope islands, while plural existential scope-taking, like singular existential scope-taking, is not.

Notice in closing that it is impossible for δ to apply to operators like *if*, since the distributivity operator only distributes over a predicate's *lexical* arguments. In a sense, this improves on QR-based treatments of exceptional scope-taking, which have to make stipulations about the distribution of distributivity operators, since they could sensibly occur under an exceptionally QR'd plural indefinite. Because we do not use QR for scope-taking, we do not spuriously create new predicates in the syntax for δ to target.

4.5.2 Multiple indefinites

Two indefinites inside a single scope island can take scope in different ways beyond the boundaries of the scope island. Sentence (4.18a) naturally admits a reading entailing that for some relative of mine x, if x draws up a favorable will, I'll inherit a house; to bring the reading out, imagine continuing the discourse with *she lives in Highland Park*. This reading requires the first indefinite, *a relative of mine*, to take exceptional scope out of the antecedent of the conditional, and the second indefinite, *a favorable will*, to take scope inside the antecedent of the conditional. Symmetrically, sentence (4.18b) readily admits a reading entailing that for some relative of mine x, if some persuasive lawyer or other visits x, I'll inherit a house; again, to bring out the reading, imagine continuing with e.g. *she lives in Highland Park*. This reading requires the first indefinite, *a persuasive lawyer*, to take scope inside the antecedent of the conditional, and the second indefinite, *a relative of mine*, to take exceptional scope outside the antecedent of the conditional.

(4.18) a. If ⟨a relative of mine draws up a favorable will⟩, I'll inherit a house.
b. If ⟨a persuasive lawyer visits a relative of mine⟩, I'll inherit a house.

I'll refer to this phenomenon as *selective exceptional scope-taking*. This section details how the account of exceptional scope works to derive these cases.

At first it might appear that the semantics only allows for *un*selective exceptional scope-taking. Consider a case like (4.18b). We might expect that a fully evaluated meaning for the scope island *a persuasive lawyer visits a relative of mine* would have to look as follows:

$$\textbf{a.law}_x \multimap \textbf{a.rel}_y \multimap (\text{visits } y\ x)^{\eta}$$

All this Mt program gives us is one big agglomerated mass of nondeterminism: instead of two indefinites, there's a single undifferentiated mass of indefiniteness. Thus, though we may certainly re-Lift this fully evaluated meaning into a tower (see immediately below), the only exceptional scope derivations we'll be able to construct will assign both indefinites the same scope relative to any operators outside the island. We will not be able to derive selective exceptional scope cases like

(4.18a) and (4.18b).

$$\begin{aligned}
(\textbf{a.law}_x \multimap \textbf{a.rel}_y \multimap (\textsf{visits}\ y\ x)^\eta)^\uparrow &= \frac{(\textbf{a.law}_x \multimap \textbf{a.rel}_y \multimap (\textsf{visits}\ y\ x)^\eta)_p \multimap [\,]}{p} && \uparrow \\
&= \frac{\textbf{a.law}_x \multimap (\textbf{a.rel}_y \multimap (\textsf{visits}\ y\ x)^\eta)_p \multimap [\,]}{p} && \text{Assoc} \\
&= \frac{\textbf{a.law}_x \multimap \textbf{a.rel}_y \multimap (\textsf{visits}\ y\ x)^\eta_p \multimap [\,]}{p} && \text{Assoc} \\
&= \frac{\textbf{a.law}_x \multimap \textbf{a.rel}_y \multimap [\,]}{\textsf{visits}\ y\ x} && \text{LeftID}
\end{aligned}$$

In fact, the account as it stands actually does predict selective exceptional scope-taking, though it is true that evaluating scope islands to type-Mt programs will not do. The explanation rests on the polymorphism of $\uparrow$ and $\multimap$ (see Fact 3.12), which allows us to compose fully evaluated higher-order programs meanings of type MMt with more structure than a program of type Mt can accommodate. The type of $\uparrow$ and $\multimap$ is given as $\mathsf{M}\alpha \to (a \to \mathsf{M}\beta) \to \mathsf{M}\beta$. If we apply $\uparrow$ to *a relative of mine*, setting α to e and β to Mt (instead of t), we have the following:

$$\left(\begin{array}{c} \mathsf{M}e \\ \\ \text{a relative of mine} \\ \\ \textbf{a.rel} \end{array}\right)^{\uparrow} = \begin{array}{c} \dfrac{\mathsf{MM}t}{e} \\ \\ \text{a relative of mine} \\ \\ \dfrac{\textbf{a.rel}_x \multimap [\,]}{x} \end{array}$$

An application of internal Lift (this time setting the inner result type to Mt) brings us to a three-level tower with **a.rel** on the third level. This allows us to derive the sought-after reading of (4.18b). We begin by composing up the scope island: in addition to fiddling with **a.rel** as just detailed, we Lift and then externally Lift the subject, Lift *visits* twice, and compose everything up via two instances of higher-order combination:

$$\left(\begin{array}{c} \dfrac{\dfrac{\mathsf{MM}t}{\mathsf{M}t}}{e} \\ \\ \text{a lawyer} \\ \\ \dfrac{\dfrac{[\,]}{\textbf{a.law}_x \multimap [\,]}}{x} \end{array} \left(\begin{array}{cc} \dfrac{\dfrac{\mathsf{MM}t}{\mathsf{M}t}}{e \to e \to t} & \dfrac{\dfrac{\mathsf{MM}t}{\mathsf{M}t}}{e} \\ \\ \text{visits} & \text{a rel of mine} \\ \\ \dfrac{\dfrac{[\,]}{[\,]}}{\textsf{visits}} & \dfrac{\dfrac{\textbf{a.rel}_y \multimap [\,]}{[\,]}}{y} \end{array}\right)\right) = \begin{array}{c} \dfrac{\dfrac{\mathsf{MM}t}{\mathsf{M}t}}{t} \\ \\ \text{a lawyer visits a rel of mine} \\ \\ \dfrac{\dfrac{\textbf{a.rel}_y \multimap [\,]}{\textbf{a.law}_x \multimap [\,]}}{\textsf{visits}\ y\ x} \end{array}$$

Lowering the resulting three-level tower means applying regular Lower (i.e. $\downarrow$) both to the bottom two levels and to the tower as a whole. The inner Lower turns a two-level tower into a program of type Mt, and the outer Lower ultimately yields a program of type MMt. Notice that the type MMt qualifies as fully evaluated per Definition 4.1 since all continuation arguments have been saturated. This means we've satisfied our obligations relative to the scope island.

$$\left(\frac{\textbf{a.rel}_y \multimap [\,]}{\left(\dfrac{\textbf{a.law}_x \multimap [\,]}{\text{visits } y\ x}\right)^{\downarrow}}\right)^{\downarrow} = \left(\frac{\textbf{a.rel}_y \multimap [\,]}{\textbf{a.law}_x \multimap (\text{visits } y\ x)^{\eta}}\right)^{\downarrow} \qquad \downarrow$$
$$= \underbrace{\textbf{a.rel}_y \multimap (\textbf{a.law}_x \multimap (\text{visits } y\ x)^{\eta})^{\eta}}_{\text{MM}t} \quad \downarrow$$

Crucially, however, the result this time is *not* an undifferentiated mass of nondeterminism. We have an extra layer of structure relative to the previous derivation, and this extra structure allows us to unfold the program by retracing the steps we took to evaluate it. An initial application of Lift brings us to a two story tower...

$$\left(\textbf{a.rel}_y \multimap (\textbf{a.law}_x \multimap (\text{visits } y\ x)^{\eta})^{\eta}\right)^{\uparrow} = \frac{(\textbf{a.rel}_y \multimap (\textbf{a.law}_x \multimap (\text{visits } y\ x)^{\eta})^{\eta})_m \multimap [\,]}{m} \qquad \uparrow$$
$$= \frac{\textbf{a.rel}_y \multimap (\textbf{a.law}_x \multimap (\text{visits } y\ x)^{\eta})^{\eta}_m \multimap [\,]}{m} \qquad \text{Assoc}$$
$$= \frac{\textbf{a.rel}_y \multimap [\,]}{\textbf{a.law}_x \multimap (\text{visits } y\ x)^{\eta}} \qquad \text{LeftID}$$

...and second application of Lift, inside the tower, brings us back to where we began pre-evaluation, i.e. a three-level tower with **a.rel** scoping on the third level and **a.law** on the second (I omit the full details of the inner Lifting since it parallels cases we've seen several times already):[9]

$$\frac{\textbf{a.rel}_y \multimap [\,]}{(\textbf{a.law}_x \multimap (\text{visits } y\ x)^{\eta})^{\uparrow}} = \frac{\textbf{a.rel}_y \multimap [\,]}{\dfrac{\textbf{a.law}_x \multimap [\,]}{\text{visits } y\ x}}$$

In contrast with the previous derivation, the two indefinites live on different scopal levels post-Reset. This means they can take post-evaluation scope in different ways, in particular allowing us to derive the *a relative of mine > if > a persuasive lawyer* reading of example (4.18b) .

Symmetrically, we can reverse the scope of the subject and object indefinites in order to derive a case like (4.18a) (i.e. by applying internal Lift to the subject and external Lift to the object). Both indefinites still survive Reset, free to take selective exceptional scope beyond the boundaries of the scope island:

$$\underbrace{\textbf{a.rel}_x \multimap (\textbf{a.fav.will}_y \multimap (\text{draws.up } y\ x)^{\eta})^{\eta}}_{\text{MM}t}$$

9 Note that the type of the post-Reset tower may differ from the type of the pre-Reset tower.

In sum, we predict three (substantively different) ways to layer the alternatives/issues in cases with two indefinites inside a scope island—one undifferentiated M*t* program, and two distinct MM*t* programs corresponding to two possible relative scopes of the two indefinites on the island—and we exploit these structures to give an account of selective exceptional scope. Moreover, we can do this with as many indefinites as we like: a sentence with three indefinites can yield a fully evaluated MMM*t*—with as many monadic layers as we'd need to differentiate all the indefinites post-evaluation. Exceptional scope is thus predicted to be selective in general.

Is there any evidence, independent of exceptional scope-taking, that the prediction the grammar makes is a good one? If there can be higher-order structure in alternative sets, as I am suggesting, can we see the effects of this anywhere else? I think the answer is yes. For one, an analogous claim about the existence/empirical utility of higher-order nondeterministic meanings has been motivated in the closely related domain of questions under the rubric of *higher-order questions* Dayal 1996, 2002, Fox 2012.[10] For another: AnderBois 2010, Barros 2013 argue that sluicing (e.g. *John met a linguist, but I don't know who*, see e.g. Chung et al. 1995, Merchant 2001) is "anaphora to issues", in the sense that the meaning of the antecedent clause has to be isomorphic to the meaning of the sluiced clause. Inquisitive semantics (a close relative of the semantics here, see e.g. Groenendijk & Roelofsen 2009 and Section 5.5) models issues as nondeterministic sets of propositions, very much in line with the nondeterministic semantics for indefinites on offer here.

If sluicing requires an isomorphic nondeterministic structure and a sentence with two indefinites is *3-ways ambiguous* in its possible nondeterministic structures, then we predict that there will be three possibilities for sluicing in cases where the discourse serves up a clause with two indefinites. And in fact this turns out to be the case. The sluice in (4.19a) is (at least) 2-ways ambiguous: the *who* in the sluiced clause can be heard as targeting beaters or beat-ees (some of my informants, including me, can hear it as 3-ways ambiguous, with a reading on which the *who* targets beater-beat-ee pairs); and the multiple sluice in (4.19b) offers evidence for yet a third nondeterministic structuring of the first sentence.[11]

(4.19) a. Someone$_i$ beat someone$_j$ up the other day, but I can't remember who$_{i/j}$.
b. Someone$_i$ beat someone$_j$ up the other day, but I can't remember who$_i$ whom$_j$.

Though the existence of multiple sluicing in English is rather controversial and the data not too pristine (e.g. Lasnik 2014), languages with uncontroversial multiple sluicing (and richer case systems) bear out the full paradigm. Here's a Hungarian example due to Anna Szabolcsi (p.c.):

(4.20) Valaki	megvert	valakit	de	nem	tudom	hogy	{ki, kit, ki kit}.
someone	up-beat	someone-Acc	but	not	know-1sg	SUBORD	{who, whom, who whom}

'Someone beat someone up, but I don't know {who, whom, who whom}.'

10 Notice in this respect that if we treat wh-words as nondeterministic (as in e.g. Hamblin 1973, Kratzer & Shimoyama 2002), we likewise predict their insensitivity to islands and derive the existence of higher-order questions.

11 NB: something like ∃-closure over outputs can't be what distinguishes the relevant issues. Closure over outputs obliterates alternatives and should accordingly render anaphora to the closed-over issue impossible, contrary to fact, e.g. *someone$_i$ beat someone$_j$ up the other day, but I don't know who$_j$, or why he$_i$ thought it was a good idea.*

Let's suppose that question words have basically the same semantics as indefinites (see AnderBois 2010 for more details than I can give here). In (4.19) and (4.20), the subject-targeting sluice will be isomorphic to a $\mathsf{MM}t$ program with the subject indefinite's effects taking widest scope, the object-targeting sluice will be isomorphic to a $\mathsf{MM}t$ program with the object indefinite's effects taking widest scope, and the subject-object-targeting sluice will be isomorphic to a $\mathsf{M}t$ program with the subject's and object's indefiniteness agglomerated.

In sum, Lifted indefinites presuppose very little about the type of object they scope over (i.e. they're *polymorphic* on their scope). They just ask us to do whatever it is we were going to do anyway a number of times (once for each individual satisfying the restrictor). If you were planning on returning a boolean, the indefinite asks you to do that a bunch of times. If you were planning on returning a fancier value, i.e. a $\mathsf{M}t$, the indefinite just asks you to do *that* a bunch of times. Importantly, quantifiers like *every linguist* and *no linguist don't* work this way since they go looking for boolean values inside of their scope in order to return a truth condition as a value; they demand their result type be $\mathsf{M}t$, never the higher-order $\mathsf{MM}t$.

4.5.3 Exceptional scope out of relative clauses

Selective exceptional scope-taking also comes in handy for analyzing exceptional scope out of relative clauses. An earlier such example was *no student$_i$ read every book that a teacher of his$_i$ recommended*. The challenge posed by examples of this sort is that the side effects of the indefinite and the extraction gap need to be differentiable post-evaluation. Here is the reason: since the relative clause gap's side effects must end up within the scope of the determiner *every* in order to be bound, it follows that if the indefinite and the gap's side effects scope on the same level, the indefinite must also end up within the scope of *every*. Thus, exceptional scope of indefinite side effects out of relative clauses calls for *selective* exceptional scope-taking out of the relative clause.

Just as two indefinites can take selective exceptional scope out of islands, we can differentiate the side effects incurred by the indefinite and the gap post-evaluation. Here I'll analyze the slightly simpler relative clause *a teacher recommended*. We begin by deriving a three-level tower in (4.21). We exploit the polymorphism inherent in Lifting to fix the outermost return type to $\mathsf{MM}t$ (as we fix the second-layer return types to $\mathsf{M}t$). After combination, the indefinite's effects live on the third level, and gap's on the second.

(4.21)
$$\left(\begin{array}{c} \begin{array}{c} \mathsf{MM}t \\ \hline \mathsf{M}t \\ \hline e \end{array} \\[1ex] \text{a teacher} \\[1ex] \begin{array}{c} \textbf{a.teach}_x \multimap [\;] \\ \hline [\;] \\ \hline x \end{array} \end{array} \left(\begin{array}{cc} \begin{array}{c} \mathsf{MM}t \\ \hline \mathsf{M}t \\ \hline e \to e \to t \end{array} & \begin{array}{c} \mathsf{MM}t \\ \hline \mathsf{M}t \\ \hline e \end{array} \\[1ex] \text{recommended} & \textsc{gap} \\[1ex] \begin{array}{c} [\;] \\ \hline [\;] \\ \hline \mathsf{recd} \end{array} & \begin{array}{c} [\;] \\ \hline \textbf{pro}_y \multimap [\;] \\ \hline y \end{array} \end{array}\right)\right) = \begin{array}{c} \begin{array}{c} \mathsf{MM}t \\ \hline \mathsf{M}t \\ \hline t \end{array} \\[1ex] \text{a teacher recommended } \textsc{gap} \\[1ex] \begin{array}{c} \textbf{a.teach}_x \multimap [\;] \\ \hline \textbf{pro}_y \multimap [\;] \\ \hline \mathsf{recd}\ y\ x \end{array} \end{array}$$

As with the multiple-indefinites derivations in the previous section, Lowering now happens in two parts. See (4.22). In the first step we discharge the inner Lower. In the second step we discharge the outer Lower. The result is a fully evaluated program with type MMt:

(4.22)
$$\begin{aligned}\left(\frac{\textbf{a.teach}_x \multimap [\,]}{\left(\dfrac{\textbf{pro}_y \multimap [\,]}{\text{recd}\ y\ x}\right)^{\downarrow}}\right)^{\downarrow} &= \left(\frac{\textbf{a.teach}_x \multimap [\,]}{\textbf{pro}_y \multimap (\text{recd}\ y\ x)^{\eta}}\right)^{\downarrow} && \downarrow \\ &= \underbrace{\textbf{a.teach}_x \multimap (\textbf{pro}_y \multimap (\text{recd}\ y\ x)^{\eta})^{\eta}}_{\mathsf{MM}t} && \downarrow\end{aligned}$$

As with the multiple indefinites case, post-evaluation we can pull apart this higher-order program into a fully-articulated three-story tower, where the indefinite's side effects life on the third level, and the gap's side effects live on the second. Of course, a universal quantifier has no such freedom. As in tensed clauses, its quantificational force dies with the evaluation of the relative clause.

4.6 Disjunction

4.6.1 Exceptionally scoping disjunction

Rooth & Partee 1982 (see also Schlenker 2006, Brasoveanu & Farkas 2011) point out that disjunction readily takes existential scope out of scope islands while (unstressed) conjunction does not:

(4.23) a. Bill hopes that someone will hire a maid or a cook. ($\vee$ > hopes)
(Rooth & Partee 1982, ex. 21c)

b. Bill hopes that someone will hire a maid and a cook. (*$\wedge$ > hopes)
(Rooth & Partee 1982, ex. 21a)

Intriguingly, disjunctions don't respect the BRC. As Rooth & Partee point out, (4.23a) has a reading with disjunction scoping over *hopes*, but where where *maid* and *cook* are interpreted *de dicto*—i.e. with the indefinites *inside* the scope of *hopes*. On that reading, Bill either hopes *de dicto* that someone will hire a maid (i.e. the hoped-for proposition is "someone hires a maid"), or he hopes *de dicto* that someone will hire a cook. Notice that we can replace e.g. *maid* with *wizard* on the indicated reading of (4.23a) without thereby committing ourselves to the existence of wizards.

Along similar lines, (4.24a) can be interpreted as follows: either Anna said everyone submitted his vita, or she said everyone submitted his portfolio (i.e. with the disjunction scoping over *everyone*, but with each of the disjuncts bound by *everyone*). And (4.24b) can be interpreted with the disjunction taking scope over *everyone*, but with each of the disjuncts taking scope under *everyone*: either everyone ate a steak, or everyone ate a hamburger (i.e. with disjunction scoping over *everyone*, but each of the disjoined indefinites within the scope of *everyone*; note also in this vein that the *de dicto* wide-scope-disjunction reading of (4.23a) has the disjoined indefinites in the scope of *hopes*).

(4.24) a. Anna either said everyone$_i$ submitted his$_i$ vita or his$_i$ portfolio. (I've forgotten which.)

b. Either everyone ate a steak or a hamburger. (I've forgotten which.)

This contrasts with the behavior of indefinites outlined in Section 4.2.3. Here, the existential scope of disjunction evidently shouldn't be thought of in terms of the entire disjunction scoping to the disjunctive operator's scope position: standard treatments of disjunction and quantifier raising will require the disjuncts to be interpreted at the disjunction's scope position.[12]

Importantly, just like indefinites, disjunctions license donkey anaphora (e.g. Rooth & Partee 1982, Groenendijk & Stokhof 1991, Stone 1992, Fiengo & May 1994). For instance, (4.25a) has a reading meaning that whenever I see John, I scream John's name, and whenever I hear Bill, I scream Bill's name. Something similar holds for (4.25b), which has a reading requiring every farmer to beat every donkey or horse he owns. Examples (4.25c) and (4.25d) show that the phenomenon is cross-categorial. Note that the disjoined constituent does not itself have to be the donkey-antecedent, as (4.25a), (4.25c), and (4.25d) show (a point which Stone 1992 emphasizes and argues to make trouble for some dynamic treatments of donkey anaphora[13]).

(4.25) a. Whenever I see John or hear Bill, I scream his name.
b. If a farmer owns a donkey or a horse, he beats it.
c. If Mary waves at John or Bill, so does Sue.
d. Whenever Max uses the fax or Oscar uses the Xerox, I can't.

To round out the paradigm, I note that: (i) Exceptional existential scope of disjunctions feeds anaphora. The indicated indexing of (4.26a) is illicit when the disjunction scopes within its minimal tensed clause, but licit when the disjunction scopes out of its minimal tensed clause (and has donkey truth conditions when the disjunction takes intermediate exceptional scope, i.e. that every student who read every book written by Chomsky cited Chomsky, and every student who read every book written by May cited May); (ii) A disjoined plural can take existential scope out of a scope island, but it cannot take distributive scope out of a scope island, see (4.26b), despite the fact that disjunctive plurals license distributive interpretations within their minimal tensed clause, see (4.26c). Both behaviors are in line with how indefinites behave in these sorts of configurations.

(4.26) a. Every student who read every book written by [Chomsky or May]$_i$ cited him$_i$.
b. If my aunts or my uncles die, I'll inherit a fortune. ($*\delta >$ if)
c. Either my aunts or my uncles are going to write me a check. ($\delta > \exists$)

Perhaps indefinites and disjunctions have something in common which allows them to both take exceptional scope and bind singular pronouns. In fact, Alonso-Ovalle 2006 argues on independent grounds (specifically, with respect to counterfactual conditionals) that disjunctions introduce sets of alternatives (i.e. have a nondeterministic semantics). See also Zimmermann 2000, Aloni 2007, Groenendijk & Roelofsen 2009, who argue for nondeterministic analyses of disjunctions on independent grounds with reference to free-choice phenomena and questions. The nondeterministic

12 A proposal in terms of QR is conceivable but complicated. It will require unrestricted type-Lifting, generalized disjunction, binding reconstruction, and two varieties of QR: one which leaves a higher-order trace and one which leaves a type-e trace.

13 Editorial: Stone's argument has been consistently mis-interpreted in the literature as showing that *all* dynamic treatments of donkey anaphora are problematic. In fact it shows only that dynamic disjunction has to be externally dynamic, as Stone himself pointed out.

treatment of disjunction parallels how indefinites are modeled in dynamic and alternative semantics. The following section shows that a nondeterministic semantics allows disjunctions, like indefinites, to extend their existential import out of scope islands and bind donkey pronouns, while at the same time deriving how indefinites and disjunctions differ with respect to the BRC.

4.6.2 Program disjunction

I begin by defining a nondeterministic notion of *program disjunction* for programs in the State.Set monad in Definition 4.6, and I use program disjunction to give *or* a scopal, nondeterministic semantics in Definition 4.7. The effect of program disjunction of m and n in the State.Set monad is, given an input stack s, to collect the value-stack pairs output by m at s and the value-stack pairs output by n at s into a single nondeterministic program. The effect of **or** is to thusly collect the result of evaluating two scopal programs.

Definition 4.6 (Program disjunction).

$$m \sqcup n := \lambda s.\, m\, s \cup n\, s$$

Definition 4.7 (Disjunction).

$$\frac{\mathsf{M}\beta}{\alpha} \to \frac{\mathsf{M}\beta}{\alpha} \to \frac{\mathsf{M}\beta}{\alpha}$$

$$\text{or}$$

$$\textbf{or} := \lambda mnk.\, m\, k \sqcup n\, k$$

An example of how this works for the sentence *Chomsky or May left* is given in (4.27) below. The result is the program disjunction of a meaning corresponding to *Chomsky left* and a meaning corresponding to *May left*. This in turn is equivalent to sequencing the nondeterministic part of the program (here, $\mathsf{c}^{\eta\triangleright} \sqcup \mathsf{m}^{\eta\triangleright}$) with the deterministic part of the program.

(4.27)
$$\left(\begin{array}{c} \dfrac{\mathsf{M}t}{e} \\[2ex] \text{Chomsky} \\[2ex] \dfrac{\mathsf{c}_x^{\eta\triangleright} \multimap [\,]}{x} \end{array} \left(\begin{array}{cc} \dfrac{\mathsf{M}t}{e} \to \dfrac{\mathsf{M}t}{e} \to \dfrac{\mathsf{M}t}{e} & \dfrac{\mathsf{M}t}{e} \\[2ex] \text{or} & \text{May} \\[2ex] \textbf{or} & \dfrac{\mathsf{m}_x^{\eta\triangleright} \multimap [\,]}{x} \end{array}\right) \begin{array}{c} \dfrac{\mathsf{M}t}{e \to t} \\[2ex] \text{left} \\[2ex] \dfrac{[\,]}{\mathsf{left}} \end{array}\right)$$

$$\begin{array}{rr} \rightsquigarrow & (\mathsf{c}^{\eta\triangleright} \multimap (\mathsf{left}\, x)^{\eta}) \sqcup (\mathsf{m}^{\eta\triangleright} \multimap (\mathsf{left}\, x)^{\eta}) \ \downarrow \\ = & (\mathsf{c}^{\eta\triangleright} \sqcup \mathsf{m}^{\eta\triangleright})_x \multimap (\mathsf{left}\, x)^{\eta} \ \downarrow \end{array}$$

On the last line, the program to the left of the sequencing operator is a nondeterministic program equivalent to the indefinite *someone identical to Chomsky or May*. Because disjunctive programs have an indefinite-like flavor, we predict that they will be able to bind donkey pronouns in sentences such as *if I see Chomsky or May, I wave to him*, in the same way donkey-binding by an indefinite worked for *if someone walked, she ran* (see Fact 3.4).

Because the result of evaluating a disjunction is a nondeterministic State.Set program, we predict that the side effects triggered by a disjunction will survive Reset. This entails that disjunctions are capable both of binding across sentence boundaries (see Schlenker 2011 for data that support this prediction), and of taking exceptional scope over higher operators:

$$\begin{aligned}
((\mathsf{c}^{\eta\triangleright} \sqcup \mathsf{m}^{\eta\triangleright})_x \multimap (\mathsf{left}\,x)^{\eta})^{\uparrow} &= \frac{((\mathsf{c}^{\eta\triangleright} \sqcup \mathsf{m}^{\eta\triangleright})_x \multimap (\mathsf{left}\,x)^{\eta})_p \multimap [\,]}{p} && \uparrow \\
&= \frac{(\mathsf{c}^{\eta\triangleright} \sqcup \mathsf{m}^{\eta\triangleright})_x \multimap (\mathsf{left}\,x)^{\eta}_{p} \multimap [\,]}{p} && \text{Assoc} \\
&= \frac{(\mathsf{c}^{\eta\triangleright} \sqcup \mathsf{m}^{\eta\triangleright})_x \multimap [\,]}{\mathsf{left}\,x} && \text{LeftID}
\end{aligned}$$

As a final point, Stone 1992 emphasizes that a disjunction can provide an antecedent for a donkey pronoun even in cases where the donkey pronoun is anaphoric to proper subparts of each disjunct, as in cases like *whenever I see Alf$_i$ or hear Cal$_i$, I scream his$_i$ name*, and points out that this creates trouble for some dynamic treatments of donkey anaphora. Our account has no difficulty with these cases. Consider the following derivation:

(4.28)
$$\begin{array}{c}\frac{\mathsf{M}t}{e} \\[1ex] \text{I} \\[1ex] \frac{[\,]}{\mathsf{me}}\end{array}
\left(\begin{array}{c}\frac{\mathsf{M}t}{e\to t} \\[1ex] \text{see Alf} \\[1ex] \frac{\mathsf{a}_x^{\eta\triangleright} \multimap [\,]}{\mathsf{see}\,x}\end{array}
\left(\begin{array}{c}\frac{\mathsf{M}t}{e\to t} \to \frac{\mathsf{M}t}{e\to t} \to \frac{\mathsf{M}t}{e\to t} \\[1ex] \text{or} \\[1ex] \textbf{or}\end{array}
\begin{array}{c}\frac{\mathsf{M}t}{e\to t} \\[1ex] \text{hear Cal} \\[1ex] \frac{\mathsf{c}_x^{\eta\triangleright} \multimap [\,]}{\mathsf{hear}\,x}\end{array}\right)\right)$$

$$\leadsto \quad (\mathsf{a}_x^{\eta\triangleright} \multimap (\mathsf{see}\,x\,\mathsf{me})^{\eta}) \sqcup (\mathsf{c}_x^{\eta\triangleright} \multimap (\mathsf{hear}\,x\,\mathsf{me})^{\eta}) \quad \downarrow$$

Even as we disjoin constituents larger than the Bind-shifted bits, the resulting disjunctive program hosts a dref in a state of nondeterministic superposition between Alf and Cal, ready and willing to nondeterministically fix the reference of downstream pronouns.

4.6.3 Disjunction and the BRC

In seeming contravention of the BRC, a disjunction can scope above some operator, while its disjuncts scope below that operator—e.g. on the wide-scope-disjunction, narrow-scope-indefinites reading of *everyone ate a steak or a burger*). This is explained by appealing to disjunction's inherent *polymorphism* (Barker 2002; polymorphic disjunction distinct from *generalized* disjunction à la

Rooth & Partee 1982, Partee & Rooth 1983). If we externally Lift two indefinites and apply the disjunction to the resulting three-level towers, as in (4.29), we end up with program disjunction scoping over c, while c scopes over the indefinites. If *everyone* is externally Lifted, it will scope over the indefinites (by virtue of being to their left), but will fail to scope over the nondeterminism introduced by disjunction (since that nondeterminism lives on the third level).

(4.29)
$$\left(\begin{array}{c}\begin{array}{c}\mathsf{M}t\\\hline \mathsf{M}t\\\hline e\end{array}\\[2ex] \text{a steak}\\[2ex] \begin{array}{c}[\,]\\\hline \textbf{a.steak}_x \multimap [\,]\\\hline x\end{array}\end{array}\left(\begin{array}{c}\begin{array}{c}\mathsf{M}t\\\hline \mathsf{M}t\\\hline e\end{array}\to\begin{array}{c}\mathsf{M}t\\\hline \mathsf{M}t\\\hline e\end{array}\to\begin{array}{c}\mathsf{M}t\\\hline \mathsf{M}t\\\hline e\end{array}\\[2ex] \text{or}\\[2ex] \textbf{or}\end{array}\begin{array}{c}\begin{array}{c}\mathsf{M}t\\\hline \mathsf{M}t\\\hline e\end{array}\\[2ex] \text{a burger}\\[2ex] \begin{array}{c}[\,]\\\hline \textbf{a.burger}_x \multimap [\,]\\\hline x\end{array}\end{array}\right)\right)$$

$$= \quad \lambda c.\, c\left(\frac{\textbf{a.steak}_x \multimap [\,]}{x}\right) \sqcup c\left(\frac{\textbf{a.burger}_x \multimap [\,]}{x}\right)$$

Cases where a disjunction takes wide scope but an *anaphoric* effect takes narrow scope (e.g. *Anna either said every candidate$_i$ should submit his$_i$ vita or his$_i$ portfolio*) work similarly, with c out-scoping **pro**, but program disjunction out-scoping c:

$$= \quad \lambda c.\, c\left(\frac{\textbf{pro}_x \multimap [\,]}{\textsf{vita}\ x}\right) \sqcup c\left(\frac{\textbf{pro}_x \multimap [\,]}{\textsf{portfolio}\ x}\right)$$

It is also possible to generate wide-scope disjunction, wide-scope quantification readings for sentences like *someone ate every steak or every burger* (i.e. the reading on which either every steak was eaten by someone or other, or every burger was eaten by someone or other). The key difference between this derivation and the previous two is that we've *internally* Lifted *every steak* and *every burger* so that their scopal side effects live on the top level of the tower. If *someone* is externally Lifted, this yields the correct reading.

(4.30)
$$\left(\begin{array}{c}\begin{array}{c}\mathsf{M}t\\\hline \mathsf{M}t\\\hline e\end{array}\\[2ex] \text{every steak}\\[2ex] \begin{array}{c}\textbf{ev.steak}\,(\lambda x.\,[\,])\\\hline [\,]\\\hline x\end{array}\end{array}\left(\begin{array}{c}\begin{array}{c}\mathsf{M}t\\\hline \mathsf{M}t\\\hline e\end{array}\to\begin{array}{c}\mathsf{M}t\\\hline \mathsf{M}t\\\hline e\end{array}\to\begin{array}{c}\mathsf{M}t\\\hline \mathsf{M}t\\\hline e\end{array}\\[2ex] \text{or}\\[2ex] \textbf{or}\end{array}\begin{array}{c}\begin{array}{c}\mathsf{M}t\\\hline \mathsf{M}t\\\hline e\end{array}\\[2ex] \text{every burger}\\[2ex] \begin{array}{c}\textbf{ev.burger}\,(\lambda x.\,[\,])\\\hline [\,]\\\hline x\end{array}\end{array}\right)\right)$$

$$= \quad \lambda c.\left(\textbf{ev.steak}\left(\lambda x.\, c\,\frac{[\,]}{x}\right)\right) \sqcup \left(\textbf{ev.burger}\left(\lambda x.\, c\,\frac{[\,]}{x}\right)\right)$$

4.6.4 Higher-order programs

As with indefinites, the polymorphism of disjunction automatically predicts the existence of higher-order disjunctive programs. For example, in a case with two disjunctions, we will allow three substantively distinct layerings, as follows:

$$(\pi \sqcup \theta)^\eta \sqcup \psi^\eta$$
$$\pi^\eta \sqcup (\theta \sqcup \psi)^\eta$$
$$\pi \sqcup \theta \sqcup \psi$$

Here is an example derivation of *Mary or John or Bill* which gives a layering corresponding to the first of these options, i.e. one whose nondeterministic structure (abstracting away from State) is $\{\{\mathsf{m}, \mathsf{j}\}, \mathsf{b}\}$. We disjoin two-level towers for *Mary* and *John*, Lift the result into a three-level tower, and then disjoin the resulting three-level tower with a three-level tower for *Bill*. As with the higher-order derivation of indefinites, we finish by doing an inner application of Lower to collapse the bottom two stories, then a final application of Lower to give something with type $\mathsf{MM}t$:

(4.31)
$$\left(\begin{array}{c}\frac{\mathsf{MM}t}{\frac{\mathsf{M}t}{e}}\\[1em] \text{Mary or John}\\[1em] \frac{[\,]}{\lambda k.\, k\,\mathsf{m} \sqcup k\,\mathsf{j}}\end{array}\left(\begin{array}{ccc}\frac{\mathsf{MM}t}{\frac{\mathsf{M}t}{e}} \to \frac{\mathsf{MM}t}{\frac{\mathsf{M}t}{e}} \to \frac{\mathsf{MM}t}{\frac{\mathsf{M}t}{e}} & & \frac{\mathsf{MM}t}{\frac{\mathsf{M}t}{e}}\\[1em] \text{or} & & \text{Bill}\\[1em] \textbf{or} & & \frac{[\,]}{\frac{[\,]}{\mathsf{b}}}\end{array}\right)\right)$$

$$\begin{array}{rrl} = & \lambda c.\, c\,(\lambda k.\, k\,\mathsf{m} \sqcup k\,\mathsf{j}) \sqcup c\,(\lambda k.\, k\,\mathsf{b}) & \mathsf{A} \\ \rightsquigarrow & \lambda c.\, c\,(\mathsf{m}^\eta \sqcup \mathsf{j}^\eta) \sqcup c\,\mathsf{b}^\eta & \downarrow \\ \rightsquigarrow & (\mathsf{m}^\eta \sqcup \mathsf{j}^\eta)^\eta \sqcup \mathsf{b}^{\eta\eta} & \downarrow \end{array}$$

As with indefinites, this higher-order structure will allow two disjunctions inside a scope island to take exceptional scope separately outside the island and suggests that we might be able to find independent motivation in sluicing for the existence of these higher-order programs. Both of these predictions seem to be borne out. As to the first, (4.32a) allows a reading meaning either my uncle or great uncle is such that if either my aunt or he dies, I'll inherit a fortune—i.e. with the latter disjunction taking exceptional scope out of the conditional, while the former disjunction scopes in situ. Notice in this respect that Rooth & Partee's 1982 example (4.33) has a disjunction taking exceptional scope out of an island, even as other sources of nondeterminism on the island (i.e. the indefinites) do not. As to the second: (4.33) is indeed 3-ways ambiguous. It does not require Steve to be refusing to say exactly *who* stole my pencil (though it has a reading which entails this); perhaps Steve knows that the thief either has to have been Mary or John, or has to have been Bill (and is refusing to say); perhaps he knows the thief either has to have been Mary, or has to have been John or Bill (and is refusing to say).[14]

(4.32) a. If my aunt or my *uncle* or *great*-uncle (I've forgotten which) dies, I'll inherit a fortune.
b. Bill hopes that someone will hire a maid or a cook

(4.33) Steve knows whether Mary or John or Bill stole my pencil, but he won't tell me which.
(3-ways ambiguous)

4.7 Comparison to other theories

I will briefly compare my theory to one classic theory of indefinites, the choice-functional theory, and one newer theory, Brasoveanu & Farkas's 2011 account based on *Independence-Friendly Logic* (e.g. Hintikka 1973, Väänänen 2007). The difficulties these theories face are representative of a wider class of pseudo-scope treatments of the scope-taking of indefinites, i.e. treatments which allow the grammar to assign indefinites apparent scope over an operator without the indefinite (or its side effects) actually scoping over that operator.

4.7.1 Choice functions

Reinhart 1997, Winter 1997, Kratzer 1998, Chierchia 2001, Kratzer 2003, and others suggest that the exceptional scope properties of indefinites are best explained by using *choice functions*.[15] A choice function is any f such that for any set P, $f\,P \in P$; e.g. if f is a choice function, $f\,\mathsf{ling}$ is a linguist. The idea is that choice functions are used to interpret indefinites, and that choice functions can be *existentially bound* at arbitrary points in a tree (interpreting indefinites as free type-e variables existentially bound at arbitrary distances was convincingly shown by Reinhart 1997 to make a number of unacceptable predictions; choice functions were introduced in part to sidestep these issues). This derives the exceptional scope properties of indefinites; since the indefinite can acquire "scope" without movement, it can "scope" out of islands without moving out of them:

(4.34) $\exists f\,.\,f$ (relative of mine) dies $\Rightarrow$ I'll inherit a house.
$\approx$ there's a way to choose a relative of mine such that if s/he dies, I'll inherit a house

Pseudo-scope mechanisms such as these—i.e. where an indefinite becomes independent of some operator without needing to move over that operator—make a number of bad predictions. Schwarz 2001 points out that when a downward-entailing operator binds into the argument of a choice function,

14 Thanks to Anna Szabolcsi for discussion of the data.

15 Schwarzschild 2002 proposes a closely related account in terms of *singleton domain restrictions* (see von Fintel 2000 for pertinent discussion). The criticisms of the choice-functional theory detailed in this section carry over there.

the resulting truth conditions are too weak; (4.35) is incorrectly predicted to have a reading which only requires no candidate to have submitted *every* paper he wrote (e.g. f might pick every candidate's worst paper). This is a BRC violation: existentially closing the choice function outside the scope of an operator that binds into the indefinite makes a reading available that's not derivable by scoping the indefinite itself over that operator.

(4.35) No candidate$_i$ submitted a paper he$_i$ wrote.
$\rightsquigarrow \exists f.$ no candidate x submitted f (paper x wrote)
$\approx$ there's a way to choose papers by candidates such no candidate submitted *that* paper of his

We can observe BRC violations even when the operator that binds into the indefinite isn't decreasing. Geurts 2000 points out that, because choice functions are *functions*, in any case where it's known that the girls all fancy the same boys, (4.36) is incorrectly predicted to have a reading on which there is a *single* boy that every girl fancies. (Given the same set again and again, the choice function must always select the same element from it).

(4.36) Every$_i$ girl gave a flower to a boy she$_i$ fancied.
(Geurts 2000, ex. 5)
$\rightsquigarrow \exists f.\, \forall x.\, \textsf{girl}\, x \Rightarrow \textsf{gave-a-flower-to}\, (f\, (\lambda y.\, \textsf{boy}\, y \wedge \textsf{fancied}\, y\, x))\, x$

Relatedly, choice functions make bad predictions for layered DPs in cases like (4.37). Perversely, we're able to existentially close the choice function used to interpret the entire object indefinite at a point higher than an operator that scopes over the choice function used to interpret *the embedded indefinite*. The predicted truth conditions are bizarre and sharply ungrammatical—roughly that every famous linguist wrote a book I didn't read (i.e. assuming no two famous linguists wrote the same book). Something like this is impossible with genuine scope-taking: in QR terms, it requires moving *a famous linguist* out of its container DP, and them moving the trace-hosting remnant back over *a famous linguist*, thereby un-binding the trace (my account of this datum was given in Section 4.4.3).

(4.37) I didn't read a book by a famous linguist.
$\rightsquigarrow \exists f \neg \exists f'.\, \textsf{read}\, (f\, (\textsf{book-by}\, (f'\, \textsf{famous-linguist})))\, \textsf{me}$
$\approx \forall x.\, \textsf{famous-linguist}\, x \Rightarrow \exists y.\, \textsf{book-by}\, x\, y \wedge \neg \textsf{read}\, y\, \textsf{me}$

Using choice functions for exceptional scope also leads to bad predictions for disjunctions. Consider a sentence like *Anna either told me no candidate should submit his$_i$ vita or his$_i$ portfolio (but I've forgotten which)*. We'd like, recall, to derive truth conditions on which what Anna told me was either that no candidate should submit his vita, or that no candidate should submit his portfolio. A natural thing to try on the choice-functional approach is treating the disjunction as introducing a choice function over a set ranging over the values of the disjuncts (cf. Hagstrom 1998, Cable 2010, Slade 2011). Then we existentially close the disjunction choice function high:

(4.38) $\exists f.\, \neg \exists x.\, \textsf{cand}\, x \wedge \textsf{submit}\, (f\, \{\textsf{vita}\, x,\ \textsf{portfolio}\, x\})\, x$

These are not the right truth conditions. To get the reading we're after, we need to guarantee that f uniformly picks a vita or a portfolio from any vita-portfolio doubleton. This is not what (4.38) says: f might pick v_1 from $\{\mathsf{v}_1, \mathsf{p}_1\}$ and p_2 from $\{\mathsf{v}_1, \mathsf{p}_2\}$. The truth conditions are thus that no candidate should submit both his vita and his portfolio. This is not a possible reading of the sentence.

Finally, choice functions are an awkward fit with anaphora. While basic cross-sentential anaphora is not too difficult to account for—one might suppose choice functions are combined with dynamic interpretation, and that the individual selected by a given choice function can end up as a dref and dynamically bind a pronoun in a subsequent sentence—this leaves it unexplained why exceptional scope-taking *feeds* cross-sentential anaphora, as in cases like (4.39).

(4.39) If ⟨a relative of mine$_i$ dies⟩, I'll inherit a house. I only wish I knew who she$_i$ was.

The indicated indexing is licit when (and only when) *a relative of mine* takes exceptional scope out of the antecedent of the conditional. On this reading, *she* obligatorily ranges over (female) relatives of mine. Since the choice-functional account of the exceptional scope reading here leaves the descriptive content *relative of mine* in situ, it is a mystery on dynamic treatments how that descriptive content ends up accessible to the pronoun *she*.[16]

4.7.2 Independence-Friendly Logic

Another, recent treatment of exceptional scope is given in Brasoveanu & Farkas 2011 ('B&F'), drawing on Independence-Friendly Logic (e.g. Hintikka 1973, Väänänen 2007), as well as work on plural anaphora within dynamic semantics (van den Berg 1996, Nouwen 2007, Brasoveanu 2007, 2008). On B&F's theory, indefinites are superscripted with sets containing the variables whose values they can covary with. If a variable v is not among an indefinite's superscript set, the indefinite's denotation is required to be *independent* of v (and thus of any quantifiers that bind v), in the specific sense that the indefinite's value cannot vary per choice of v. Thus, a structure like (4.40a) assigns the indefinite *some problem in Binding Theory* scope over the universal quantifier over papers but under the universal quantifiers over students. A structure like (4.40b) gives the indefinite scope over both universals. (I am fudging a bit on the details; B&F actually offer a semantics for a regimented first-order-logic-like language).

(4.40) a. Every student$_x$ read every paper$_y$ ⟨written about some$^{\{x\}}$ problem in Binding Theory⟩.
b. Every student$_x$ read every paper$_y$ ⟨written about some$^{\{\ \}}$ problem in Binding Theory⟩.

In order to extend this treatment to indefinites taking exceptional scope over negation, B&F must posit an intensional, quantificational semantics for negation and allow indefinites to be indexed to world variables bound by operators such as negation.

If this is all that is said, nothing rules out a structure like (4.41), which requires the indefinite to be independent of a universal that binds into the indefinite's restrictor—i.e. this structure incorrectly

16 See Chierchia 2005, Brennan 2011 for an attempted fix combining dynamic binding of choice functions with an E-type treatment of pronouns (e.g. Heim 1990, Elbourne 2005), and Section 5.2.2 for a reason not to go this way.

derives a BRC violation. For this reason, B&F stipulate that an indefinite's restrictor must be interpreted relative to the same set of the variables at which the indefinite as a whole is interpreted. See Brasoveanu & Farkas 2011, ex. 36 (the fact that the stipulation can be made at all counts as a point in favor over choice-functions-based analyses of exceptional scope). While this suffices to rule out (4.41), it does not do so in an explanatory way.

(4.41) Every candidate$_x$ submitted a$^{\{\,\}}$ paper he$_x$ wrote.

B&F must also stipulate a "No Skipping Constraint" to rule out problematic configurations such at (4.42) and (4.43): (4.42) requires the indefinite to simultaneously scope above the higher universal and below the lower universal (whose scope is restricted to a position below the higher universal by virtue of containing a pronoun bound by the higher universal), and (4.43) requires the indefinite to be interpreted above *most students* but below *every professor* (whose scope is confined to its minimal tensed clause). Neither of these meanings seem to track possible readings of their respective sentences. No Skipping requires, inter alia, that indefinites be interpreted relative to *sequences* of variables in lieu of sets in order to avoid ordering paradoxes such as these.

(4.42) Every student$_x$ read every paper$_y$ he$_x$ heard that some$^{\{y\}}$ professor had recommended.

(4.43) Most students$_x$ noticed that ⟨every professor$_y$ recommended a$^{\{y\}}$ paper about scope⟩.
Brasoveanu & Farkas 2011, ex. 72

These unattested readings are impossible to derive in my semantics. Because exceptional scope-taking reflects side effects taking bona fide scope, it's nonsensical to talk about an indefinite's side effects scoping under a quantifier Q while simultaneously scoping over some higher quantifier which scopes over Q.

In order to derive the exceptional scope properties of disjunction, B&F suggest treating disjunctions as existential quantifiers restricted by their individual disjuncts. It is not clear how to extend this to handle cases where individual disjuncts can scope below a disjunction. For one, if the disjuncts stay in situ, the relevant reading would seem to violate the prohibition on an indefinite's restrictor having access to the variables outside the indefinite's superscript. See (4.44) and (4.45). For another, even if admitted, neither (4.44) nor (4.45) represents the reading of interest: (4.44) requires there to be a single burger-or-steak token that every boy ate, and (4.45) requires there to be a single vita-or-portfolio token that no candidate should submit.

(4.44) Every boy$_x$ ate [a$^{\{x\}}$ steak or a$^{\{x\}}$ burger]$^{\{\,\}}$

(4.45) Anna either said no candidate$_x$ should submit [his$_x$ vita or his$_x$ portfolio]$^{\{\,\}}$

Finally, eschewing bona fide scope-taking for exceptionally-scoping indefinites makes it difficult to see how exceptional scope-taking feeds anaphora in cases where a dynamically closed operator such as negation scopes over the scope island. In such cases, the indefinite and all its descriptive content remain in situ and should become inaccessible for anaphora.

4.7.3 Previous continuations-based work

Barker & Shan 2008 propose a continuations-based theory that attempts to derive donkey anaphora from the exceptional scope properties of indefinites (and disjunctions). The rough idea is that donkey pronouns are in fact in-scope bound by a static scope-taker (e.g. the $(e \to t) \to t$ generalized quantifier $\lambda k.\, \exists x.\, \mathsf{ling}\, x \wedge k\, x$), but that the sort of scope-taking required is of a sort only available to indefinites (and disjunctions). In previous work (Charlow 2010; see also Barker & Shan 2014 for a detailed discussion of the issues), I've pointed out a number of difficulties with trying to assimilate donkey anaphora to in-scope binding. For this reason, I've concluded that donkey anaphora requires a semantics that recognizes both state-changing and nondeterministic side effects, i.e. the State.Set semantics I've proposed here.

I briefly touch on another issue. Barker & Shan's account relies in a crucial way on indefinites being able to take scope in ways that quantifiers such as *every linguist* cannot. However, no model of how indefinites or disjunctions (or any other exceptional scope-takers, see Chapter 5) do this is proposed. While Barker 2002 implicitly uses a Reset to prevent quantifiers from scoping out of islands, that account of scope islands ends up trapping everything, including indefinites, on the island. In contrast, my account leverages a single property of indefinites (and disjunctions)—that they incur nondeterministic and state-changing side effects—to explain donkey anaphora and exceptional scope-taking in one bound.

4.8 Conclusion

Any theory of scope-taking must make some provisions for scope islands. A natural proposal in a denotational theory of scope-taking is to require that scope islands be evaluated. As this chapter argued in detail, this necessary stipulation derives the exceptional scope properties of indefinites, along with an intricate pattern of data that took us from plurals to disjunction to sluicing and back.

The gains I've reported here are modular in the sense that they're compatible with insights and techniques gleaned from other theories of dynamic interpretation. Though, I assumed a minimal, stack-based DyS-style side effects regime (following Dekker 1994), any proposal that associates indefinites with a nondeterministic semantics—from Heim 1982 and Groenendijk & Stokhof 1991 to Muskens 1996 and Brasoveanu 2007—can be adapted into a theory based on side effects that makes precisely the same predictions as the underlying State.Set monad.[17]

I note in closing that there's a real sense in which exceptional scope is due not to (e.g.) an indefinite *itself* taking exceptional scope, but rather to the indefinite-hosting *scope island* taking scope. In other words, the way an indefinite acquires scope over constituents outside a scope island is fundamentally *indirect*. Even as the indefinite's scope-taking is confined to the island, the island inherits the indefinite's side effects and passes them to some larger computation. Intriguingly, similar proposals have been made for apparent island violations in the closely related domain of wh-quantification (Dayal 1996, 2002). This connection should ultimately be explored further.

17 My account is also compatible with a wide range of assumptions about semantic combination; we can for instance readily accommodate essentially conjunctive interpretation schemes (e.g. Parsons 1990).

Chapter 5

Generalized exceptional scope

5.1 Introduction

This chapter takes the ideas developed in the previous three and uses them to offer exceptional-scope-based insight into other problem domains. We will look at four cases. Sections 5.2 and 5.3 explore predictions of our theory of exceptional scope-taking from the point of view of anaphoric side effects persisting and taking scope after evaluation. Section 5.2 looks at what I call surprising sloppy readings, cases where we have reason to suppose that e.g. a proper name like *Bill* can bind a pronoun from an arbitrarily deeply embedded position. Section 5.3 extends the treatment of dynamically closed quantifiers to cover maximal discourse reference in examples such as *exactly one linguist left; she was tired* and argues that the effects of exceptional anaphoric scope-taking can be observed for maximal drefs. Section 5.4 offers an account of the island-insensitivity and selectivity of association with focus by reformulating Rooth's 1985 alternative semantics in terms of the present theory of side effects and scope. Finally, Section 5.5 gives, novelly, a compositionally and empirically robust formulation of Kratzer & Shimoyama's 2002 alternative semantics, a possible revision of the account in Chapter 4 for theorists who wish to use the account of exceptional scope but not the dynamic account of anaphora (in so doing, we illustrate the modularity of the proposal on offer; the difference between the two theories will turn on whether the underlying monad is Reader.Set or State.Set).

5.2 Surprising sloppy readings and cross-categorial drefs

This section uses sloppy identity in ellipsis to motivate the idea that ostensibly scope-less things like proper names and VPs nevertheless do take exceptional scope. The argument goes like this: (i) sloppy identity in ellipsis reveals the presence of binding relationships; (ii) sloppy identity is possible in cases where binding requires the antecedent for an elided sloppy pro-form to take scope out of a scope island; ergo (iii) sloppy identity reveals that antecedents for sloppy identity may take exceptional scope in the same way as indefinites. It is shown that the data falls out from the semantics already proposed: the anaphoric side effects triggered by a proper name (or anything else to which the Bind-shifter $\rhd$ applies) naturally survive evaluation, and are free to take exceptional scope post-evaluation. The conclusion is that exceptional scope-taking is a far broader phenomenon than previously recognized. In addition to the quantificational and anaphoric effects seen with indefinites and disjunction, its presence can be diagnosed via purely anaphoric phenomena, as well.

5.2.1 Sloppy readings in ellipsis: data and theoretical background

Constituents can sometimes go unpronounced when they have a salient antecedent in the discourse, as in e.g. (5.1) (CAPS in examples indicates obligatory intonational prominence, or *focus*, and ~~striked out text~~ is intended to designate the relevant interpretation of the silent material in a theory-neutral way). Call this phenomenon *ellipsis*. *Sloppy identity* in ellipsis occurs when the interpretation of the elliptical material seems to differ from the interpretation of its antecedent. Example (5.2) has a reading on which the antecedent VP seems to mean something like *likes John's mom*, and the elided VP something like *likes Bill's mom*. (The second conjunct is in fact ambiguous between the sloppy identity reading and a *strict* reading on which Bill likes John's mom. I'll more or less ignore strict readings.) We'll call the locus of the meaning shift in the ellipsis site (here, the pronoun *him*$_j$) the sloppy pro-form.

(5.1) John jumped, and then BILL did ~~jump~~.

(5.2) John$_i$ likes his$_i$ mom, and BILL$_j$ does ~~like his$_j$ mom~~ TOO.

An extensively motivated constraint on ellipsis is that a sloppy pro-form must be *bound*. This idea has its roots in Keenan 1971, Sag 1976, Williams 1977, where it was assumed that ellipsis of XP is licensed only when there is a salient antecedent XP such that antecedent and elided XPs are semantically identical. For these authors, the semantic identity requirement meant a sloppy pronoun had to be bound *within* its elided XP. In e.g. (5.2), both antecedent and elided VPs might denote $\lambda x.\,\textsf{likes}\,(\textsf{mom}\,x)\,x$—the property of liking one's own mother. In such a scenario, antecedent and elided VPs have the same meaning, ellipsis is licensed, and a sloppy interpretation is derived.

The idea that a sloppy pro-form has to be bound within the elided XP was eventually weakened on the basis of sloppy identity in *re-binding* configurations, e.g. (5.3) (see e.g. Evans 1988, Rooth 1992a, Jacobson 1992, Fiengo & May 1994, Fox 1999, Schwarz 2000, Takahashi & Fox 2005, Hartman 2011). In re-binding examples, a sloppy pro-form is necessarily bound only *outside* of the elided XP. For instance, the smallest domain in which the sloppy pronoun in (5.3) can be bound is the VP headed by *says*, which dominates the elided VP.

(5.3) John says Sue likes him, but BILL$_j$ says she DOESN'T ~~like him$_j$~~.

To account for data of this sort, theories of ellipsis licensing replaced semantic identity with a condition requiring there to be constituents reflexively dominating antecedent and elided XPs which form a *coherent discourse pair*, with coherence defined in terms of a focus-semantic notion of Contrast (e.g. Rooth 1992a,b, Heim 1997, Fox 1999, Schwarz 2000, Jacobson 2004, Takahashi & Fox 2005, Hartman 2011, Charlow 2008, 2012, Kennedy 2014).[1] I will abstract away from the

1 YP *reflexively dominates* XP iff (i) YP dominates XP or (ii) XP = YP.

particulars of focus interpretation and Contrast, though we will return to focus in Section 5.4. For now, an informal definition suffices to illustrate the point:

Definition 5.1. Contrast holds of two trees $\mathcal{S}$ and $\mathcal{T}$ only if there's a way to replace the focused elements in $\mathcal{T}$ (i.e. those in CAPS) which yields a structure $\mathcal{T}'$ such that $[\![\mathcal{S}]\!] = [\![\mathcal{T}']\!]$.

Importantly, Contrast is satisfied in sloppy elliptical constructions (re-binding or no) *only when the sloppy pro-form is bound*. E.g. if the elided *him* in (5.3) is bound, its interpretation necessarily *covaries* with the interpretation of its antecedent. Thus, replacing the stressed *BILL* with *John* yields a $\mathcal{T}'$ meaning that John said Sue likes John. This is the meaning of the first conjunct, and Contrast is satisfied. On the other hand, if the elided *him* isn't bound, replacing *BILL* with *John* will fail to have any effect on *him*'s interpretation; $\mathcal{T}'$ means that John says Sue doesn't like *Bill*, and Contrast is not satisfied. In short, Contrast-based theories of ellipsis allow us to use the availability of sloppy interpretations as a way to diagnose the presence of binding.

When are binding and sloppy ellipsis possible? Though it's often assumed *X* can bind *Y* only if *X* c-commands *Y* at surface structure (see e.g. Reinhart 1983, Büring 2005, Barker 2012 for discussion and references), sloppy identity cases (along with other data, see e.g. Barker 2012 for an extensive review, along with the discussions of non-c-command anaphora in the previous chapters) suggest that this picture is too restrictive. The second sentence in example (5.4) can easily mean that Bill's mom doesn't like *Bill*— i.e. with a sloppy reading of the elided pronoun—even though *Bill* doesn't c-command the sloppy pro-form on the surface. Rooth 1992a analyzed this case within a static semantics (i.e. one without dynamic binding) by supposing that covert movement could feed binding: QRing *BILL* to a position c-commanding the elided sloppy pronoun at LF allows the sloppy pronoun to be bound and Contrast to be satisfied. However, using QR to feed binding is less plausible for other cases. Example (5.5) readily admits a sloppy interpretation, on which the latter cop didn't insult Bill, but *BILL* is separated by a scope island (the tensed relative clause) from a position where it could c-command and statically bind the elided sloppy pronoun.

(5.4) John$_i$'s mom likes him$_i$. BILL$_j$'s mom DOESN'T ~~like him$_j$~~.

(5.5) The cop who arrested John$_i$ insulted him$_i$.
The cop who ⟨arrested BILL$_j$⟩ DIDN'T ~~insult him$_j$~~.
(after Wescoat 1989, as reported by Dalrymple et al. 1991)

(5.6) The cop who arrested a protestor$_i$ insulted him$_i$.

For cases similar to (5.5), Schwarz 2000 bit the bullet and proposed that island-disrespecting QR was possible and could feed ellipsis. As with indefinites, a theory of this kind requires us to simply stipulate that certain constituents can QR out of islands, and others cannot. Other researchers proposed that *Bill$_j$* binds the elided sloppy pronoun *him$_j$* in (5.5), much as *a protestor* binds *it* in (5.6) (Tomioka 1999, Hardt 1999, Charlow 2012). The assumption was that *donkey-binding* configurations can satisfy Contrast and license sloppy ellipsis, much as in-scope binding does.

5.2.2 Surprising sloppy readings and previous accounts

Whether donkey-anaphoric configurations can satisfy Contrast in fact depends on the semantics assumed for donkey pronouns. For instance, these data cause trouble for E-type theories of donkey anaphora (Heim 1990, Tomioka 1999, Elbourne 2005, Büring 2004). On E-type theories, donkey pronouns are treated as definite descriptions with some associated descriptive content. For instance, Elbourne's theory of donkey pronouns entails that the elliptical clause in example (5.5) has essentially the same LF and interpretation as *the cop who arrested BILL DIDN'T ~~insult him Bill~~*, with *Bill* treated as a predicate and *him* as a definite determiner. This prevents Contrast satisfaction for the same reason that a lack of binding did: varying the focused antecedent *BILL* does not have the effect of varying the denotation of the pronoun. For Contrast to be satisfiable, there needs to be a lowest common denominator covert description for the antecedent and elliptical instances of *him*. In the general case, this description may need to be as semantically bleached as *the thing* (Leu 2005), and it is thus a significant challenge to construct an E-type theory loose enough to avoid under-generating sloppy readings but restricted enough to avoid over-generating (Charlow 2011).

Dynamic theories of anaphora fare better. In dynamic accounts, the relationship between a donkey antecedent and a donkey pronoun is a genuine binding relationship, and so the interpretation of a donkey pronoun covaries with the interpretation of its antecedent. Thus, on a dynamic theory, replacing *BILL* with *John* in (5.5) gives a $\mathcal{T}'$ semantically indistinguishable from the antecedent (see Charlow 2012 for a dynamic formalization of the Contrast condition).

Yet even dynamic binding of sloppy pronouns is not always straightforward to achieve. We can construct cases analogous to (5.5) but with the sloppy pronoun's putative antecedent embedded arbitrarily deep. Regardless of the relative embedding, sloppy anaphora remains possible, even in cases where analyses couched in terms of donkey anaphora predict that the sloppy pronoun cannot be bound. See (5.7a). The sloppy reading is grammatical, but *BILL* is separated by a scope island (its minimal tensed clause) from the nearest position from which it could donkey-bind the elided sloppy pro-form—i.e. above the dynamically closed *everyone*. Something similar holds for (5.7b), where *Pete* is separated by a scope island (the relative clause) from the nearest position from which it could donkey bind the elided sloppy pronoun—i.e. again above the dynamically closed *everyone*. I'll call cases of sloppy anaphora which seem to require exceptional scope of the antecedent for the sloppy pro-form instances of *surprising sloppy readings*.

(5.7) a. If everyone thinks John will come, we'll have to invite him$_i$.
If everyone thinks ⟨BILL will ~~come~~⟩, we WON'T ~~have to invite him$_j$~~.

b. If everyone who despises Walt^i comes, I'll feel bad for him$_i$.
If everyone who ⟨despises PETE$_j$⟩ does ~~come~~, I WON'T ~~feel bad for him$_j$~~.

5.2.3 Solution: unbounded anaphoric scope

I've been emphasizing that existential scope-taking can feed anaphora. The possibility of sloppy identity in (5.7a) and (5.7b) thus leads us to conclude that proper names, like indefinites, are subject to anaphora-feeding existential scope-taking. Though proper names, VPs, and NPs are ostensibly scope-less, their exceptional existential scope-taking nevertheless feeds binding of the elided pro-forms, which allows for the sloppy readings observed here.

Given the semantics developed in Chapters 2 and 3, and the theory of exceptional scope detailed in Chapter 4, we *automatically* predict that the dref associated with e.g. a proper name can take exceptional scope in exactly the same way as an indefinite since it survives Reset (cf. Fact 4.1). As with an indefinite, Reset has no effect on the dref generated by an application of a Bind-shift:

$$\begin{aligned}
\left(\frac{\mathsf{b}_x^{\eta\rhd} \multimap [\,]}{\mathsf{come}\ x}\right)^{\downarrow\uparrow} &= (\mathsf{b}_x^{\eta\rhd} \multimap (\mathsf{come}\ x)^{\eta})^{\uparrow} && \downarrow \\
&= \frac{(\mathsf{b}_x^{\eta\rhd} \multimap (\mathsf{come}\ x)^{\eta})_p \multimap [\,]}{p} && \uparrow \\
&= \frac{\mathsf{b}_x^{\eta\rhd} \multimap (\mathsf{come}\ x)_p^{\eta} \multimap [\,]}{p} && \text{Assoc} \\
&= \frac{\mathsf{b}_x^{\eta\rhd} \multimap [\,]}{\mathsf{come}\ x} && \text{LeftID}
\end{aligned}$$

The prediction, then, is that the drefs induced by proper names, like the nondeterminism and drefs induced by indefinites, freely scope upward. This solves the issue with (5.7a) and (5.7b). In (5.7a), the dref corresponding to Bill may take exceptional inverse scope over *everyone*:

(5.8)
$$\left(\begin{array}{c} \frac{\frac{\mathsf{M}t}{\mathsf{M}t}}{t} \\[2ex] \text{everyone} \\[2ex] \frac{\frac{[\,]}{\mathbf{eo}\,(\lambda x.\,[\,])}}{x} \end{array} \left(\begin{array}{cc} \frac{\frac{\mathsf{M}t}{\mathsf{M}t}}{t} & \frac{\frac{\mathsf{M}t}{\mathsf{M}t}}{t} \\[2ex] \text{thinks} & \text{BILL will } \text{~~come~~} \\[2ex] \frac{\frac{[\,]}{[\,]}}{\mathsf{thinks}} & \frac{\frac{\mathsf{b}_y^{\eta\rhd} \multimap [\,]}{[\,]}}{\mathsf{come}\ y} \end{array}\right)\right) = \begin{array}{c} \frac{\frac{\mathsf{M}t}{\mathsf{M}t}}{t} \\[2ex] \text{everyone thinks BILL will ~~come~~} \\[2ex] \frac{\frac{\mathsf{b}_y^{\eta\rhd} \multimap [\,]}{\mathbf{eo}\,(\lambda x.\,[\,])}}{\mathsf{thinks}\ (\mathsf{come}\ y)\ x} \end{array}$$

$$\rightsquigarrow \quad \mathsf{b}_y^{\eta\rhd} \multimap (\forall x.\,\mathsf{thinks}\ (\mathsf{come}\ y)\ x)^{\eta} \quad \Downarrow,\ \mathbf{eo}$$

Once $\mathsf{b}^{\eta\rhd}$ acquires exceptional scope over **eo**, it is free to bind the pronoun. Moreover, because this is a genuine binding relationship, replacing the focused expressions *BILL* and *WON'T* with *John* and *will* (and running the same derivation) yields something with an identical meaning to *if everyone thinks John$_i$ will come, we'll have to invite him$_i$*, and CONTRAST is predicted satisfiable.

$$\begin{aligned}
&\mathbf{if}\ (\mathsf{b}_y^{\eta\rhd} \multimap (\forall x.\,\mathsf{thinks}\ (\mathsf{come}\ y)\ x)^{\eta})\ (\mathbf{pro}_z \multimap (\neg\mathsf{invite}\ z\ \mathsf{we})^{\eta}) \\
&\qquad\qquad = ((\forall x.\,\mathsf{thinks}\ (\mathsf{come}\ \mathsf{b})\ x) \Rightarrow \neg\mathsf{invite}\ \mathsf{b}\ \mathsf{we})^{\eta} \quad \mathbf{if}
\end{aligned}$$

Unlike indefinites, this exceptional scope-taking scope has no “quantificational”/nondeterministic import. Yet surprising sloppy readings, which *require* binding, diagnose its presence.

5.2.4 Cross-categorial sloppy readings

Sloppy readings are cross-categorial (e.g. Hardt 1999, Schwarz 2000). Example (5.9) can mean that when John has to clean, he doesn’t want to *clean*. This looks like a garden-variety donkey-anaphoric sloppy case, analogous to (5.5). This motivates the idea that ellipsis resolution is a matter of *anaphora* resolution—i.e. that the antecedent VP has the form *want to P*, with *P* a VP-type pro-form that can shift its interpretation in ellipsis just as pronouns do (e.g. Szabolcsi 1992, Hardt 1993, 1999, Schwarz 2000, Charlow 2008, 2012, Barker 2013).

(5.9) When John has to cook, he doesn’t want to ~~cook~~.
When he has to CLEAN, he doesn’t ~~want to clean~~ either.

Surprising sloppy readings also turn out to be cross-categorial, see (5.10) (cf. Hardt 1999, Charlow 2012, Hardt et al. 2013). Though this case allows a sloppy interpretation (i.e. if everyone thinks you’ll come to office hours, you need to *come to office hours*), *come to office hours* is separated by a scope island from a scope position above a dynamically closed operator (i.e. *everyone*).

(5.10) If everyone thinks you’ll do the reading, you don’t need to ~~do the reading~~.
If everyone thinks ⟨you’ll come to office hours⟩, you DO ~~need to come to office hours~~.

There is also a subtler issue of accessibility. In both (5.9) and (5.10), the antecedent for the *text-level* VP ellipsis in the first sentence (e.g. *want to P need to P*) is in a position inaccessible to the second sentence (since both the antecedents and consequents of conditionals are dynamically closed). This seems to require some exceptional scope of e.g. *want to P* outside the conditional. See Hardt 1999, Charlow 2012, and especially Hardt et al. 2013 for discussion.

Our theory generalizes to cross-categorial surprising sloppy readings. As a first step, notice that the polymorphic Bind-shifter ► already predicts the possibility of *higher-order drefs*—for instance, drefs that bottle up a tower. See Fact 5.1. Moreover, allowing pro-forms to be polymorphic means that pronouns can be *anaphoric to towers*. See Fact 5.2.

Fact 5.1 (Generalized drefs).

$$\left(\begin{array}{c} f\,[\,] \\ \hline g\,[\,] \\ \hline a \end{array}\right)^{\blacktriangleright} = \begin{array}{c} f\left[\left(\begin{array}{c} g\,[\,] \\ \hline a \end{array}\right)^{\eta\triangleright}_{m} \multimap [\,]\right] \\ \hline m \end{array}$$

Fact 5.2 (Generalized pro-forms). When defined:

$$\textsf{pro}^{\uparrow} = \begin{array}{c} \textsf{pro}_{m} \multimap [\,] \\ \hline m \end{array} = \begin{array}{c} \textsf{pro}_{m} \multimap [\,] \\ \hline m\,(\lambda a.\,[\,]) \\ \hline a \end{array}$$

Let's see how higher-order drefs allow us to derive a basic case of sloppy identity. We will analyze *John*$_i$ *likes his*$_i$ *mom, and BILL*$_i$ *does* ~~*likes his*$_j$ *mom*~~ *TOO*. We begin by composing up *likes his mom*. Call the resulting two-level tower π.

$$\begin{array}{c} \begin{array}{c} \mathsf{M}t \\ \hline e \to t \end{array} \\[1.5em] \text{likes his mom} \\[1.5em] \left.\begin{array}{c} \mathbf{pro}_w \multimap [\,] \\ \hline \mathsf{likes}\,(\mathsf{mom}\,w) \end{array}\right\} =: \pi \end{array}$$

Next, we externally Lift π and apply the bind shifter. This makes a higher-order dref for π. Notice in particular that while m instantiates the two-level tower in the derivation, the way that $\mathbf{pro}_w$ is resolved in m will *not necessarily* fix the reference of $\mathbf{pro}_w$ in $\pi^{\eta\triangleright}$.

$$\left(\begin{array}{c} \begin{array}{c} \mathsf{M}t \\ \hline \mathsf{M}t \\ \hline e \end{array} \\[1.5em] \text{likes his mom} \\[1.5em] \begin{array}{c} [\,] \\ \hline \pi \end{array} \end{array}\right)^{\blacktriangleright} = \begin{array}{c} \begin{array}{c} \mathsf{M}t \\ \hline \mathsf{M}t \\ \hline e \end{array} \\[1.5em] \text{likes his mom} \\[1.5em] \begin{array}{c} \pi^{\eta\triangleright}_m \multimap [\,] \\ \hline m \end{array} \end{array}$$

See the derivation in (5.11). I have assumed a silent pro-form *shhh* which stands in for the elliptical VP (for simplicity, I've also ignored *does* and *too*). I've suppressed the derivation and Reset of both conjuncts. The higher-order dref in the left conjunct survives evaluation to fix the reference of the higher-order pro-form in the right conjunct. While m is in the immediate scope of $\mathsf{j}^{\eta\triangleright}$, n is in the immediate scope of $\mathsf{b}^{\eta\triangleright}$. This means that the $\mathbf{pro}_w$ in m is resolved to John, and the $\mathbf{pro}_w$ in n to Bill. The sloppy reading is derived—even as the higher-order dref corresponding to *likes his mom* takes scope over both *John* and *BILL*!

(5.11)
$$\left(\begin{array}{c} \begin{array}{c} \mathsf{M}t \\ \hline \mathsf{M}t \\ \hline t \end{array} \\[1.5em] \text{John likes his mom} \\[1.5em] \begin{array}{c} \pi^{\eta\triangleright}_m \multimap [\,] \\ \hline \mathsf{j}^{\eta\triangleright}_x \multimap m\,(\lambda P.\,[\,]) \\ \hline P\,x \end{array} \end{array} \left(\begin{array}{cc} \begin{array}{c} \mathsf{M}t \\ \hline \mathsf{M}t \\ \hline t \to t \to t \end{array} & \begin{array}{c} \mathsf{M}t \\ \hline \mathsf{M}t \\ \hline t \end{array} \\[1.5em] \text{and} & \text{BILL } \textit{shhh} \\[1.5em] \begin{array}{c} [\,] \\ \hline [\,] \\ \hline \mathsf{and} \end{array} & \begin{array}{c} \mathbf{pro}_n \multimap [\,] \\ \hline \mathsf{b}^{\eta\triangleright}_y \multimap n\,(\lambda Q.\,[\,]) \\ \hline Q\,y \end{array} \end{array}\right)\right)$$

$$\rightsquigarrow \quad \pi^{\eta\triangleright}_m \multimap \mathsf{j}^{\eta\triangleright}_x \multimap \mathsf{b}^{\eta\triangleright}_y \multimap (\mathsf{likes}\,(\mathsf{mom}\,\mathsf{j})\,\mathsf{j} \wedge \mathsf{likes}\,(\mathsf{mom}\,\mathsf{b})\,\mathsf{b})^{\eta} \quad \downdownarrows$$

Cross-categorial sloppy readings work analogously; the only difference is that the pronoun that lives inside the higher-order dref is looking for a VP-type meaning, rather than an individual.[2]

It is interesting to notice that the theory offers an essentially identical account of *paycheck pronouns* (e.g. Engdahl 1986, Jacobson 2000a; see Hardt 1999 for another theory connecting sloppy ellipsis and paycheck pronouns). For example, (5.12) can mean that Bill hates *Bill's* mom, even as *her* seems to depend on *his mom*, i.e. Bill's, for its reference. (Again, the sentence is ambiguous and admits a "strict" reading entailing that Bill hates John's mom.)

(5.12) John$_i$ likes [his$_i$ mom]$_j$, but BILL$_k$ HATES her$_j$.

The analysis is parallel. The only difference from the sloppy identity case is that the higher-order dref is an *individual*-returning program (rather than a property-returning program). Start with a regular derivation of *his mom*, calling the resulting two-level tower π:

$$\begin{array}{c} \dfrac{\mathsf{M}t}{e} \\[1em] \text{his mom} \\[1em] \left.\dfrac{\mathbf{pro}_w \multimap [\,]}{\mathsf{mom}\ w}\right\} =: \pi \end{array}$$

As before, we create a higher-order dref bottling up π by externally Lifting the just-derived two-story tower and then applying the Bind-shift:

$$\left(\begin{array}{c} \dfrac{\mathsf{M}t}{\dfrac{\mathsf{M}t}{e}} \\[1em] \text{his mom} \\[1em] \dfrac{[\,]}{\pi} \end{array}\right)^{\blacktriangleright} = \begin{array}{c} \dfrac{\mathsf{M}t}{\dfrac{\mathsf{M}t}{e}} \\[1em] \text{his mom} \\[1em] \dfrac{\pi_m^{\eta\triangleright} \multimap [\,]}{m} \end{array}$$

A full paycheck derivation using these pieces is given in (5.13). As before, π survives evaluation qua dref and is thus accessible to the second conjunct. On the top level, the higher-order pro-form $\mathbf{pro}_n$ meets the higher-order dref π on the third level; thus n ends up bound to π. On the second tier, m (also bound to π) is in the immediate scope of $\mathsf{j}^{\eta\triangleright}$, and n is in the immediate scope of $\mathsf{b}^{\eta\triangleright}$. From the perspective of m and n, the most recently introduced dref differs, and this means that the $\mathbf{pro}_w$ in

2 One way to derive the strict reading (the one that says Bill likes John's mom) would be to avoid the $\triangleright$ shift on *BILL*. Another way would be to move *BILL*'s anaphoric effects onto a higher level, out of the way of *John*'s. Yet another would be to get a more sophisticated system with more than one pronominal index.

m is resolved to John, and the $\textbf{pro}_w$ in n to Bill. The paycheck reading is derived.

(5.13)
$$\left(\begin{array}{c} \mathsf{M}t \\ \hline \mathsf{M}t \\ \hline t \\ \\ \text{John likes his mom} \\ \\ \pi_m^{\eta\triangleright} \multimap [\,] \\ \hline \mathsf{j}_x^{\eta\triangleright} \multimap m\,(\lambda v.\,[\,]) \\ \hline \mathsf{likes}\ v\ x \end{array} \left(\begin{array}{c} \mathsf{M}t \\ \hline \mathsf{M}t \\ \hline t \to t \to t \\ \\ \text{but} \\ \\ [\,] \\ \hline [\,] \\ \hline \mathsf{and} \end{array} \begin{array}{c} \mathsf{M}t \\ \hline \mathsf{M}t \\ \hline t \\ \\ \text{BILL HATES her} \\ \\ \textbf{pro}_n \multimap [\,] \\ \hline \mathsf{b}_y^{\eta\triangleright} \multimap n\,(\lambda u.\,[\,]) \\ \hline \mathsf{hates}\ u\ y \end{array}\right)\right)$$

$$\rightsquigarrow \quad \pi_m^{\eta\triangleright} \multimap \mathsf{j}_x^{\eta\triangleright} \multimap \mathsf{b}_y^{\eta\triangleright} \multimap (\mathsf{likes}\,(\mathsf{mom}\,\mathsf{j})\,\mathsf{j} \wedge \mathsf{hates}\,(\mathsf{mom}\,\mathsf{b})\,\mathsf{b})^{\eta} \quad \downdownarrows$$

We've accomplished this without positing any pronominal ambiguity (in contrast with many other analyses of paycheck pronouns, including e.g. Engdahl's). Though polymorphic, third-person pronouns (indeed, all pro-forms) have a unitary lexical semantics.[3]

5.2.5 Structure in ellipsis?

Our account of cross-categorial surprising sloppy readings relies on treating ellipsis as an anaphoric phenomenon. However, a point commonly cited in favor of ellipsis sites being syntactically represented is that we can extract out of ellipsis sites, and so there must be structure to host the extraction traces, e.g. in cases like *I know what Bill likes and what he doesn't* (e.g. Ross 1967). However, our semantics predicts that it is perfectly possible for a higher-order dref to host a gap. Start with a derivation of a gapped VP:

$$\begin{array}{c} \mathsf{M}t \\ \hline e \to t \\ \\ \text{likes } \textsc{gap} \\ \\ \textbf{pro}_x \multimap [\,] \\ \hline \mathsf{likes}\ x \end{array}$$

Creating a higher-order dref from this tower is a matter of externally Lifting and applying the Bind-shifter. A pro-VP can retrieve this higher-order, gap-hosting dref, and we thereby predict that "extraction" out of ellipsis sites is semantically coherent (and analogous to sloppy readings).[4]

3 The analysis here is simple-minded in at least one respect. If pronouns are completely polymorphic (and dref retrieval is constrained only by the need to assemble a well-typed larger program), nothing prevents a pronoun from being anaphoric to a *quantifier* (e.g. *he* could mean *everyone*, which I take to be impossible). Thus, pronouns like *him* and *her* must in fact presuppose that the dref they retrieve is non-scopal: i.e. something of type e (e.g. p) or type $\mathsf{M}e$ (e.g. $\textbf{pro}_w \multimap (\mathsf{mom}\ w)^{\eta}$). This still allows us to derive the paycheck reading using a higher-order type-$\mathsf{M}e$ dref, but I've included the derivation above to keep things simple and to emphasize the parallel with sloppy identity in VP ellipsis.

4 See Barker 2013 for a discussion of how one might ensure that the case on the "moved" element matches the case assigned to the gap within a semantic theory of ellipsis.

Another point in favor of structured ellipses is the potential for inverse scope out of elliptical VPs. Consider (5.14), the most natural reading of which allows there to be different American and Canadian flags per embassy (Hirschbühler 1982). On syntactic theories of scope assignment, this creates a paradox: the elided universal needs to take scope over its subject but is not syntactically represented and thus not subject to scope-taking. (Data of this sort are not accounted for in other semantic accounts of ellipsis like Hardt 1993, 1999.)

(5.14) An American flag hangs in front of every embassy, and a CANADIAN flag does TOO.

Consider the derivation of *hangs in front of every embassy* in (5.14). We've jacked the universal quantifier up to the top level via an internal application of Lift. This suggests we're getting ready to construct an inverse scope derivation. Quite so, but now notice that this *three*-level tower can be transmuted into a higher-order dref (as before, all we need to do is externally Lift and apply $\rhd$).

$$\dfrac{\mathsf{M}t}{\dfrac{\mathsf{M}t}{e \to t}}$$

hangs in front of every embassy

$$\left.\dfrac{\textbf{ev.emb}\,(\lambda y.\,[\;])}{\dfrac{[\;]}{\textsf{hangs.before}\;y}}\right\} =: \pi$$

While the quantificational force of *every embassy* is, as expected, neutralized on evaluation/Reset, the higher-order *dref*, quantificational as the moment it was born, survives evaluation and is free to be passed on to the following sentence. There, it can be picked up by a pro-VP and used to generate an inverse-scoping interpretation:

$$\dfrac{\mathsf{M}t}{t}$$

an American flag hangs in front of every embassy

$$\dfrac{\pi_m^{\eta\rhd} \multimap [\;]}{\forall y.\,\textsf{emb}\;y \Rightarrow \exists x.\,\textsf{am.flag}\;x \wedge \textsf{hangs.before}\;y\;x}$$

In sum, cross-categorial sloppy readings motivate the idea that ellipsis is an anaphoric phenomenon. The existence of higher-order drefs and higher-order pro-forms is already predicted by the most unrestricted version of our semantics for dref introduction and pro-forms. The semantics also predicts that higher-order drefs will survive evaluation and take exceptional scope in the same way as the nondeterministic and anaphoric side effects triggered by indefinites/disjunctions and the anaphoric side effects triggered by ostensibly scope-less proper names. And, finally, our semantics shows that gap-hosting ellipses and inverse-scoping ellipsis constructions provide no motivation for structure in ellipsis sites (while cross-categorial sloppy readings provide strong evidence against it).

5.3 Maximal drefs and dynamic generalized quantifiers

We have so far distinguished two kinds of scopal meanings: those that return dynamically closed programs (i.e. programs built on negation), and those that don't. This over-simplifies the empirical picture. Determiners such as *every*, *no*, *more than five*, *exactly one*, and so on *do* in fact give rise to discourse anaphora, and so their semantics should not be given in terms of functions into programs that are utterly dynamically closed. This discourse anaphora generally has a flavor of *maximality* or *exhaustivity*: that is, a pronoun anaphoric to a "dynamically closed" quantifier can typically be paraphrased in terms of a definite description.

This section examines maximal drefs and dynamic generalized quantifiers. I'll propose a conservative semantics for *dynamic generalized quantifiers*—i.e. those quantifiers that give rise to maximal drefs on evaluation and examine a number of predictions this proposal makes in conjunction with the account of exceptional scope. Because DPs like *no linguist* turn out to map their scopes into programs that aren't completely dynamically closed, we predict that they will give rise to a kind of exceptional scope-taking. Crucially, the scope-taking of the anaphoric residue that remains after evaluating a quantifier of this sort yields very different results from what would obtain if the quantifier *itself* were able to take exceptional scope. I argue that these predictions are borne out.

5.3.1 Background

Two types of maximal discourse anaphora are prominent in modern dynamic theories (see e.g. Kamp & Reyle 1993, van den Berg 1996, Nouwen 2003, 2007, Brasoveanu 2007, 2008). *They* in (5.15) is naturally construed as referring to a plural individual comprised of the first-years. Call this *restrictor anaphora*. Meanwhile, in (5.16) *they* naturally refers to the senators who admire Cruz, qua plural individual, and in (5.17) *she* naturally refers to *the* linguist who left. Call this *refset anaphora*. Both restrictor and refset anaphora are kinds of maximal discourse anaphora: in each of these cases, the pronoun's meaning can be paraphrased in terms of a definite description naming *all* the individuals who so-and-so, e.g. all the senators who admire Cruz in example (5.16).[5]

(5.15) No first-years showed up. They went to another party.

(5.16) Few senators admire Cruz, and they're all very junior.

(5.17) Exactly one linguist left. She was tired.

A standard response to data of this sort is to suppose that quantifiers such as these are in fact externally dynamic, but in a different way from indefinites (ibid.). In particular, when e.g. *few senators* takes scope over a sentence, the result is not a program completely devoid of drefs. Instead, the quantifier contributes some maximal drefs of its own (while still wiping out the drefs generated

5 Schlenker 2012 argues that the notion of maximality relevant in these cases should be cashed out in terms of *informativity*, rather than in terms of the *size* of the plural individual.

within its scope). See Definition 5.2 for a schematic treatment of non-indefinite determiners in terms of a non-monadic dynamic semantics (i.e. DyS).[6] A DyS dynamic GQ denotes a function into a DyS proposition that adds maximal drefs for the individuals in its restrictor and for the individuals in its restrictor who also satisfy the nuclear scope. DET is a stand-in for the usual relation on sets of individuals associated with a given quantificational determiner $\text{DET}_{\textsf{-indef}}$, and $\mathsf{DET\,M\,N}$ corresponds to a given sentence's truth condition. An example in input-output terms is in Fact 5.3. *Exactly one linguist left* denotes a non-empty relation iff exactly one linguist left. If that's the case, the sentence outputs a dref bottling up the linguists (M), and a dref bottling up the single linguist who left (N).

Definition 5.2 (DyS dynamic GQ).

$$\begin{aligned}[\![\text{DET}_{\textsf{-indef}}\ \text{linguist(s)}]\!] \coloneqq\ &\lambda k s.\,\{\widehat{s\mathsf{MN}}\}\text{ if }\mathsf{DET\,M\,N}\text{ else }\{\,\}\\ &\text{for }\mathsf{M}\coloneqq\mathsf{ling}\\ &\text{for }\mathsf{N}\coloneqq\mathsf{ling}\cap\{x\mid\exists s'.\,s'\in k\,x\,\widehat{sx}\}\end{aligned}$$

Fact 5.3 (Input-output perspective for DyS GQs).

$$\begin{aligned}s\,[\![\text{exactly one linguist left}]\!]\,s'\text{ iff }&(\exists!x.\,\mathsf{ling}\,x\wedge\mathsf{left}\,x)\wedge s'=\widehat{s\mathsf{MN}}\\ &\text{for }\mathsf{M}\coloneqq\mathsf{ling}\\ &\text{for }\mathsf{N}\coloneqq\mathsf{ling}\cap\mathsf{left}\end{aligned}$$

Notice that the maximal dref N is available only outside the quantifier's scope: the scope argument k never sees it (nor could it, on pain of making the definition of N circular). This jibes with the datum that pronouns in the scope of a dynamic GQ are interpreted distinctly from pronouns outside the GQ's scope. In the first sentence of example (5.18), *their* can't be interpreted as anaphoric to the refset of the senators who despise their constituents; its most natural reading requires that the individuals x such that x is a senator who despises x's constituents number at least five. By contrast, the second *they* is naturally interpreted as referring to the senators who despise their constituents (it's in principle ambiguous between this construal and one that refers to the senators). See Kamp & Reyle 1993: 322–323, Nouwen 2007 for discussion.

(5.18) At least five senators despise their constituents, but they're retiring next term.

5.3.2 Dynamic GQs for anaphora

This section transposes the DyS semantics of dynamic GQs into the State.Set monad. Here, first, is a rough-and-ready characterization of how maximal discourse reference to the refset will go (this is a toy theory for the sake of illustration; the results in this section should scale up to more sophisticated semantics, cf. fn. 6). See Definition 5.3. Similar to how things worked with the toy meaning for *every/no linguist*, we output a deterministic single pair of a proposition and a stack. This time,

6 The semantics in Definition 5.2 hews closest to the approach of Kamp & Reyle 1993. Other approaches (e.g. van den Berg 1996, Nouwen 2003, 2007, Brasoveanu 2007, 2008) use states richer than stacks for reasons not closely related to our concerns at present (though everything that follows is compatible these richer structures).

however, we push an individual onto the stack containing the linguist(s) who satisfy the nuclear scope. (I'm assuming as above that dynamic GQs are interpreted distributively within their scope.)

Definition 5.3 (Dynamic GQs contributing maximal drefs).

$$\begin{array}{c} \frac{\mathsf{M}t}{e} \\[1em] \text{DET}_{\text{-indef}} \text{ linguist(s)} \\[0.5em] \lambda k s.\, \left\{ \left\langle \mathsf{DET\,M\,N},\ \widehat{s\mathsf{N}} \right\rangle \right\} \\ \quad \text{for } \mathsf{M} := \mathsf{ling} \\ \quad \text{for } \mathsf{N} := \mathsf{ling} \cap \{x \mid \exists s'.\, \langle \mathsf{True},\ s' \rangle \in k\, x\, s\} \end{array}$$

A few changes from DyS are worth noting. Most importantly, we hold onto the truth condition $\mathsf{DET\,M\,N}$ as the value properly returned by the resulting $\mathsf{M}t$, rather than treating it as a mere condition on there being any outputs at all. Second, since binding is modularized in the grammar with ►, there is no need to lexically encode that $k\,x$ is evaluated at $\widehat{sx}$, or that the restrictor is pushed onto the reference stack *by the dynamic GQ* (i.e. the restrictor can take care of itself; we'll come back to this at the end of this subsection).[7]

Here is how this semantics works to derive a case of maximal discourse anaphora to a refset. We will analyze example (5.16), namely *exactly one linguist left; she was tired*. We begin by deriving *exactly one linguist left*, in (5.19). The boolean returned after evaluation is True iff there's precisely one linguist who left, and there is a new dref generated comprising all and only the linguists who left (i.e., assuming the sentence is true, *the* linguist who left).

(5.19)
$$\left(\begin{array}{cc} \frac{\mathsf{M}t}{e} & \frac{\mathsf{M}t}{e \to t} \\[1em] \text{exactly one linguist} & \text{left} \\[1em] \frac{\textbf{ex.one.ling}\,(\lambda x.\,[\;])}{x} & \frac{[\;]}{\mathsf{left}} \end{array} \right) = \begin{array}{c} \frac{\mathsf{M}t}{t} \\[1em] \text{exactly one linguist left} \\[1em] \frac{\textbf{ex.one.ling}\,(\lambda x.\,[\;])}{\mathsf{left}\;x} \end{array}$$

$$\begin{array}{lcl} \rightsquigarrow & \textbf{ex.one.ling}\,(\lambda x.\,(\mathsf{left}\,x)^{\eta}) & \downarrow \\ = & \lambda s.\, \left\{ \left\langle |\mathsf{N}| = 1,\ \widehat{s\mathsf{N}} \right\rangle \right\} \quad \text{for } \mathsf{N} := \mathsf{ling} \cap \mathsf{left} & \textbf{ex.one.ling} \\ = & (\mathsf{ling} \cap \mathsf{left})_x^{\eta \triangleright} \multimap (|x| = 1)^{\eta} & \triangleright \end{array}$$

7 I'm bracketing the lexical entries for determiners since they're not directly relevant to the point I'm making here. (Determiners also bring up the issue of the *weak/strong* ambiguity of sentences like *exactly one man who had a quarter put it in the meter*, see e.g. Schubert & Pelletier 1989, Kanazawa 1994, Brasoveanu 2007, which I'll opportunistically ignore.)

Cross-sentential anaphora to the maximal dref is a matter of applying $\uparrow$ to the fully evaluated $\mathsf{M}t$ program and combining the result with the tower that gives the (post-Reset) meaning of *she was tired*, exactly parallel to the way a simple case of cross-sentential anaphora to a singular indefinite was derived in Chapter 4. Combination brings **pro** within the immediate scope of the maximal-dref-hosting program. This means the value returned by **pro** is, correctly, bound to the linguist who left (assuming the truth of the first conjunct; recall also that per Schwarzschild 1996 we are identifying $\{x\}$ and x).

$$\left(\begin{array}{c}\dfrac{\mathsf{M}t}{t}\\[1em] \text{exactly one linguist left}\\[1em] \dfrac{(\mathsf{ling}\cap\mathsf{left})^{\eta\rhd}_x \multimap [\,]}{|x|=1}\end{array}\left(\begin{array}{cc} \dfrac{\mathsf{M}t}{t\to t\to t} & \dfrac{\mathsf{M}t}{t}\\[1em] \text{and} & \text{she was tired}\\[1em] \dfrac{[\,]}{\mathsf{and}} & \dfrac{\mathbf{pro}_y\multimap[\,]}{\mathsf{tired}\,y}\end{array}\right)\right)$$

$$\begin{array}{lll}\leadsto & (\mathsf{ling}\cap\mathsf{left})^{\eta\rhd}_x \multimap \mathbf{pro}_y \multimap (|x|=1\wedge\mathsf{tired}\,y)^{\eta} & \downarrow\\ = & (\mathsf{ling}\cap\mathsf{left})^{\eta\rhd}_x \multimap (|x|=1\wedge\mathsf{tired}\,x)^{\eta} & \text{Binding}\end{array}$$

Evaluating the conjunction correctly leaves us with a deterministic program that returns True iff: the number of linguists who left is exactly one, and the linguist who left was tired. The updated stack is also retained post-evaluation, in case any other pronouns looking for the refset await.

As for restrictor anaphora, there is no reason (given that binding is modularized in the grammar, and drefs are naturally polymorphic) for us to suppose that the dynamic GQ goes to the trouble of putting the restrictor on the stack (the case is different for the refset since there's no constituent in the sentence corresponding to it). Instead, the restrictor itself can be analyzed as giving rise via $\blacktriangleright$ to a dref that takes exceptional scope in precisely the same way as a proper name's dref. Reference to the restrictor is thus discourse anaphora to the restrictor *qua* plural individual. Because the dref introduced by a restrictor survives Reset in the same fashion as the drefs introduced by proper names or VPs, it can be used to fix the value of pronouns in subsequent sentences.[8]

5.3.3 The upward mobility of maximal drefs

Because restrictor drefs and refset drefs survive evaluation, we predict that these drefs will be able to take scope over dynamically closed operators. This section argues that this prediction is confirmed. No matter how deeply embedded a restrictor is, it can take exceptional scope out of an island and over any number of dynamically closed operators. So it goes for the refset drefs created by dynamic GQs: the refset dref evinces exceptional scope properties, even as the quantificational scope of the

8 Clearly, this simple-minded treatment will not immediately extend to restrictor anaphora to complex nominals like *linguist who John met*, which, per Section 3.6.3, evaluate to meanings of type $e \to \mathsf{M}t$. One way to go would be to enrich the semantics of dynamic GQs, e.g. along the lines of fn. 6. Another option would be for drefs with type $e \to \mathsf{M}t$ to be interpreted as restrictors of a covert operator with the semantics of *the*, type $(e \to \mathsf{M}t) \to \mathsf{M}e$.

GQ itself is bound to its scope island.

Let's look first at restrictors. Szabolcsi 2010:102 points out that (5.20) can easily be construed as entailing the existence of a set of all the first-years (in Szabolcsi's terms, the *minimal witness* of *every first year*), even as it is impossible for *every first-year* to take distributive scope out of its minimal tensed clause. Indeed, we can observe that *every* nominal restrictor in a determiner phrase potentially has something like unbounded upward scopal mobility. That is, any of the sentences schematized in (5.21) has a reading which entails the existence of a set containing all the first-years, though in no case can the embedded quantifier over first-years take scope out of its minimal tensed clause.

(5.20) Nobody could possibly believe the rumor that ⟨every first-year got an A in phonology⟩.

(5.21) Nobody could possibly believe the rumor that ⟨{more than three, many, several, exactly ten, between three and five, most} first-years got an A in phonology⟩.

This existential scope-taking is truth-conditionally inert: if *first-year*'s world-index is the actual world (i.e. rather than belief-worlds or rumor-worlds), then its scope position is wholly irrelevant to the sentence's truth conditions. Importantly, however, this exceptional scope-taking feeds anaphora. Both (5.20) and any of the sentences schematized in (5.21) can be continued with *they're all semanticists* (i.e. the first years).[9] Though this data is unsurprising by now—the restrictor dref can take exceptional scope in precisely the same way as anything else that survives evaluation—the possibility of binding in these cases is doubly surprising for dynamic theories which locate maximal discourse reference to nominal restrictors in the semantics of the embedded quantificational determiners: for one, the quantifier is interpreted downstairs, within the scope of several dynamically closed operators, and so any drefs it outputs should remain inaccessible. For another, even if anaphoric reference to a restrictor is dissociated from the downstairs determiner, the nominal restrictor itself is still deeply embedded under several dynamically closed operators.

As for refset anaphora to dynamic GQs, we've seen time and again that the ability to survive evaluation goes hand in hand with the ability to take exceptional scope. The prediction, in other words, is that maximal drefs created by dynamic GQs should show the same exceptional scope properties as we observed for surprising sloppy readings, and that this dref can take scope separately from where the GQ's quantificational force is realized. In other words, it should be accessible outside of scope islands, even when the scope island intervenes between it and some dynamically closed operator. This prediction is confirmed, as the following cases attest:

(5.22) Everyone heard the rumor that ⟨at least six senators supported Cruz's filibuster⟩. It turned out to be erroneous: they numbered at most three.

(5.23) It's absolutely false that ⟨no senators admire Cruz⟩. But even so, they're all very junior.

In (5.22), *at least six senators* obligatorily scopes inside its minimal tensed clause (i.e below *everyone*,

9 That this is a binding phenomenon can be supported by sloppy anaphora. That is, the following is grammatical: *if nobody believes the rumor that more than three [first-years]$_i$ got an A in semantics, we'll have to tell them$_i$; if nobody believes the rumor that more than three [SECOND-years]$_i$ got an A in semantics, we WON'T ~~have to tell them$_j$~~.*

heard, and the psych predicate *rumor*)—that is, the rumor is claimed to be of the form "at least six senators support Cruz's filibuster". Nevertheless, a maximal dref bottling up the senators who admire Cruz finds its way to *they* in the subsequent sentence. Something similar holds for (5.23). Though *no senators* obligatorily scopes below *it's absolutely false that*—that is, the first sentence entails that there are senators who admire Cruz—a maximal dref bottling up the senators who admire Cruz is able to fix a value for the *they* in the next sentence.

I conclude that maximal discourse reference to the NP ∩ VP set has exceptional scope properties: it's possible to use a pronoun to refer back to a maximal dref generated by a quantifier inside of a scope island, even when a dynamically closed bit of meaning scopes over the scope island. This is predicted by my semantics. Let's see how this goes for (5.23). Assume a model where, in fact, several senators admire Cruz. Then the truth condition returned by *no senators admire Cruz*, will be False, and the outgoing stack will be extended with a plural dref comprising all and only the senators who admire Cruz. Call the resulting, fully evaluated program (type $\mathsf{M}t$) π:

$$(\textsf{sen} \cap \textsf{admire c})^{\eta\triangleright}_{x} \multimap (|x| = 0)^{\eta}$$

In 5.24, we re-Lift the evaluated scope island, apply η to the bottom story to facilitate combination with **not** (type $\mathsf{M}t \to \mathsf{M}t$), and combine everything up into a composite two-story tower. To finish off the derivation, we can apply $\uparrow$ to the bottom story of the resulting two-level tower to give a three-level tower and then Lower in one fell swoop to give a program where the value returned is a True boolean, and the plural dref of Cruz-admiring senators is available for anaphora.

(5.24)
$$\left(\begin{array}{cc} \dfrac{\mathsf{M}t}{\mathsf{M}t \to \mathsf{M}t} & \dfrac{\mathsf{M}t}{\mathsf{M}t} \\[2ex] \textsf{neg} & \text{no senators admire Cruz} \\[2ex] \dfrac{[\,]}{\textbf{not}} & \dfrac{(\textsf{sen} \cap \textsf{admire c})^{\eta\triangleright}_{x} \multimap [\,]}{(|x| = 0)^{\eta}} \end{array}\right) = \begin{array}{c} \dfrac{\mathsf{M}t}{\mathsf{M}t} \\[2ex] \textsf{neg no senators admire Cruz} \\[2ex] \dfrac{(\textsf{sen} \cap \textsf{admire c})^{\eta\triangleright}_{x} \multimap [\,]}{\textbf{not}\ (|x| = 0)^{\eta}} \end{array}$$

$$\begin{array}{rll} \rightsquigarrow & (\textsf{sen} \cap \textsf{admire c})^{\eta\triangleright}_{x} \multimap \textbf{not}\ (|x| = 0)^{\eta} & \uparrow, \Downarrow \\ = & (\textsf{sen} \cap \textsf{admire c})^{\eta\triangleright}_{x} \multimap (|x| \neq 0)^{\eta} & \textbf{not} \end{array}$$

The upshot is that negation flips the polarity of the value returned by π to True (i.e. despite the re-Lifting, the quantificational force that inhered in *no senators* pre-evaluation is going nowhere fast), while the dref M (the plural individual comprising the senators who admire Cruz) makes it out alive and is capable of binding the downstream *they* in the following sentence. A similar derivation will go through, mutatis mutandis, for examples like (5.22).

Note that any conversational context that fails to entail that there are senators who admire Cruz will cause N to be empty at some of the possibilities in the context. Per our theory of pluralities, the domain of individuals excludes the empty set, and so we predict that this sort of situation should result in a presupposition failure (e.g. Heim 1983). This seems to be a reasonable prediction. For example, this example ensures that the presupposition holds by virtue of the first sentence *it's absolutely false that no senators admire Cruz*, after which it is presumably common ground that the set of senators who admire Cruz is non-empty.

In sum, the theory developed in Chapter 4 extends in a natural way to restrictor and refset anaphora. Bind-shifted restrictors and dynamic GQs both make maximal drefs that survive evaluation; in concert with our theory of scope islands, this makes a number of good predictions. To linger on dynamic GQs for a moment, though their quantificational force is very much discharged on evaluation, a refset dref nevertheless does survive. Post-Reset, it's free to take exceptional scope in the same way as indefinites and the drefs induced by proper names and restrictors. It should be emphasized that this property is not accounted for by standard theories of dynamic GQs (e.g. van den Berg 1996, Nouwen 2003, 2007, Brasoveanu 2007): if a dynamic GQ remains in the scope of a dynamically closed operator, any drefs generated by the GQ should be wiped out by the dynamically closed operator and unavailable for anaphora beyond the scope of the dynamically closed operator. This prediction is belied by (5.22) and (5.23).[10]

5.3.4 Structural conditions on binding

The picture that emerges from surprising sloppy readings and exceptional maximal discourse reference is that binding relationships are far less constrained by the syntax than is usually supposed (even by dynamic semanticists). Specifically, the data suggests—and my theory predicts—that anaphoric side effects are *always accessible in principle*, just as nondeterministic side effects are always accessible in principle. This achieves, I submit, something closer to the full promise of dynamic semantics than standard dynamic theories offer: on my account, binding and nondeterministic scope are both truly liberated from syntax. A related perspective on the interaction of syntax and binding has been reached independently in work by Safir 2004 (see also Fiengo & May 1994). Safir's conclusions are thus consistent with—indeed, derived by—an empirically robust theory of interpretation.

These data provide additional motivation for the account of exceptional scope proposed here relative to extant treatments. Accounts of exceptional scope are generally designed so as to allow *indefinites* to project their *existential force* beyond scope islands (see Section 4.7). As such, the exceptional scope properties of proper names and maximal drefs are not explained.

5.4 Focus alternatives

We'll shift gears now, away from anaphoric side effects and to the exceptional scope properties of focus. Rooth's 1985 standard account argues that focused expressions introduce alternatives (i.e. nondeterminism) into a semantic computation. One of the key arguments Rooth gives in favor

10 It is tantalizing to think about how this sort of explanation might be extended to cover classic examples such as *it isn't the case that I don't own a radio; it's a Panasonic*. That is, perhaps indefinites, in addition to nondeterministically creating non-maximal drefs, might *also* create on evaluation a deterministic, maximal dref that takes scope separately from the indefinite's existential force, analogous to dynamic GQs. Notice for instance that double-negation cases such as this do seem to presuppose that the Panasonic is the *unique* radio owned by the speaker.

of a nondeterminstic account of focus is that a focus-sensitive operator such as *only* or *also* can associate with a focused expression or expressions *across a scope island boundary*. However, Rooth's account problematically predicts that association with focus will in general be *unselective*—i.e. that a focus-sensitive operator will unselectively quantify over all the focus alternatives it c-commands in a given interpreted structure. I offer a solution, arguing that a refactoring of Rooth semantics in terms of side effects solves both the problems of island-insensitivity and selectivity along the same lines detailed for indefinites and disjunction in Chapter 4.

5.4.1 Roothian alternative semantics

Rooth 1985, 1992b proposes a semantics for focused expressions and focus-sensitive operators such as *also* and *only* that exploits nondeterminism and nondeterministic modes of composition. Here are two important data points. First, focus placement seems to make a truth-conditional difference in the presence of focus-sensitive adverbs like *only* and *also*: sentence (5.25a) means something rather different from sentence (5.25b); the first means I'm fussy about what I imbibe, and the second means I'm fussy about what I do with Perrier. Something similar goes for *I also drink PERRIER$_F$* and *I also DRINK$_F$ Perrier*; the first entails I drink something else, and the latter that I do something else with Perrier besides drink it(?).

(5.25) a. John only drinks PERRIER$_\text{F}$.
b. John only DRINKS$_\text{F}$ Perrier.

The second key data point is that association with focus ('AWF') is *insensitive to scope islands*. Though *BILL$_F$* occurs inside of a scope island in example (5.26), it readily associates with the focus-sensitive adverb *only* sitting outside the scope island.

(5.26) I only said ⟨BILL$_\text{F}$ came⟩.

A rough-and-ready characterization of Rooth's 1985 alternative semantics is given in Definition 5.4. The core idea is that interpretation is two-dimensional. There is an interpretation function $[\![\cdot]\!]$ for assigning "regular" meanings to constituents and a second interpretation function $[\![\cdot]\!]_f$ for associating constituents with *sets of alternatives*. The basic idea behind $[\![\cdot]\!]_f$ is (i) to allow focused expressions to introduce alternatives (i.e. nondeterminism) into a semantic computation—Alt is a function from model-theoretic objects a to a set containing the (contextually relevant) alternatives to a—and (ii) to pass that nondeterminism on up the tree using nondeterministic functional application (i.e. NA from Chapter 2).

Definition 5.4 (Roothian alternative semantics).

- Focus values for non-F-marked terminals: $[\![X]\!]_f := \{[\![X]\!]\}$
- Focus values for F-marked nodes: $[\![X_\text{F}]\!]_f := \mathsf{Alt}\,[\![X]\!]$
- Focus values for non-F-marked branching nodes: $[\![X\,Y]\!]_f := \{\mathsf{A}\,x\,y \mid x \in [\![X]\!]_f \wedge y \in [\![Y]\!]_f\}$

As a simple example, the focus-semantic value associated with an utterance of *BILL$_F$ came*, i.e. $[\![\text{BILL}_\text{F} \text{ came}]\!]_f$, is $\{\mathsf{came}\, x \mid x \in \mathsf{Alt}\, \mathsf{b}\}$, a set of alternative propositions of the form "x came", with x ranging over the (contextually relevant) alternatives to Bill

Focused constituents generate focus alternatives. Focus-sensitive adverbs *quantify* over the alternatives in their sister's focus set. See the entries for *only* and *also* in Definition 5.5. *Only* requires that the only true proposition in its sister's focus set is its sister's regular value, while *also* requires that there is a true proposition in its sister's focus set besides its sister's regular value. (NB: I'm abstracting away from which entailments are asserted/presupposed, and I am treating the type t as the type of *propositions*, i.e. sets of worlds, a necessary step for deriving acceptable truth conditions.)

Definition 5.5 (Roothian focus-sensitive VP operators).

$$[\![\text{only } X]\!] \coloneqq \lambda x w.\, \{P \mid P \in [\![X]\!]_f \wedge P\, x\, w\} = \{[\![X]\!]\}$$
$$[\![\text{also } X]\!] \coloneqq \lambda x w.\, \{P \mid P \in [\![X]\!]_f \wedge P\, x\, w\} \supset \{[\![X]\!]\}$$

In addition to quantifying, focus-sensitive adverbs reset the focus value of their argument to a deterministic VP-type meaning corresponding to their regular value—e.g. $[\![\text{only } X]\!]_f = \{[\![\text{only } X]\!]\}$.

Here is an example of how nondeterministic focus alternative percolation and the semantics of focus-sensitive adverbs works to derive (5.26), a case of AWF across an island. We begin by composing up the focus value of the VP *said BILL$_F$ came*. The result is a set of properties: the NA that inheres in $[\![\cdot]\!]_f$ causes the nondeterminism triggered by the focused expression to be inherited by the larger constituent, despite the presence of the intervening island.

$$[\![\text{said } \langle \text{BILL}_\text{F} \text{ came}\rangle]\!]_f = \{\mathsf{said}\, (\mathsf{came}\, y) \mid y \in \mathsf{Alt}\, \mathsf{b}\}$$

We merge in the focus-sensitive adverb *only* and calculate the denotation of the resulting VP...

$$\begin{aligned}&[\![\text{only said } \langle \text{BILL}_\text{F} \text{ came}\rangle]\!] = \\ &\qquad \lambda x w.\, \{P \mid P \in [\![\text{said } \langle \text{BILL}_\text{F} \text{ came}\rangle]\!]_f \wedge P\, x\, w\} = \{[\![\text{said } \langle \text{BILL}_\text{F} \text{ came}\rangle]\!]\}\end{aligned}$$

...which is equivalent to the following, the property that holds of an individual x at a world w iff Bill was the only person x said came in w:

$$\lambda x w.\, \{y \mid \mathsf{said}\, (\mathsf{came}\, y)\, x\} = \{\mathsf{b}\}$$

As this example illustrates, AWF can happen at a distance without QR. NA is in effect a pseudo-scope mechanism that delivers the alternatives all the way up the tree to the nearest focus-sensitive operator.

5.4.2 Problem: selectivity

In addition to deriving the island-insensitivity of AWF, Rooth's semantics predicts that AWF will be *unselective*. That is, it predicts that a focus-sensitive operator such as *only* or *also* will obligatorily quantify over *any and all* focused expressions it c-commands in the interpreted structure (more specifically, any which have not already been interpreted by a lower c-commanding focus-sensitive operator). Here is an example, analyzing *John only introduced BILL$_F$ to SUE$_F$*, that illustrates the point. As before, we begin by deriving the focus value for the VP, a nondeterministic set of properties:

$$[\![\text{intro'd BILL}_\text{F} \text{ to SUE}_\text{F}]\!]_f = \{\textsf{intro}\ x\ y \mid x \in \textsf{Alt}\,\textsf{b} \wedge y \in \textsf{Alt}\,\textsf{s}\}$$

We merge in the focus-sensitive adverb *only* and calculate the regular denotation of the resulting VP...

$$[\![\text{only intro'd BILL}_\text{F} \text{ to SUE}_\text{F}]\!] =$$
$$\lambda zw.\,\{P \mid P \in [\![\text{intro'd BILL}_\text{F} \text{ to SUE}_\text{F}]\!]_f \wedge P\,z\,w\} = \{[\![\text{intro'd BILL}_\text{F} \text{ to SUE}_\text{F}]\!]\}$$

...which is equivalent to the following, the property that holds of an individual x at a world w iff the only introducing x did in w was of Bill to Sue.

$$\lambda zw.\,\{\langle x,\ y\rangle \mid \textsf{intro}\ x\ y\ z\ w\} = \{\langle \textsf{b},\ \textsf{s}\rangle\}$$

Thus, on the Roothian picture, *only* unselectively binds/discharges the focus alternatives in its scope (cf. e.g. Lewis 1975). But while this is certainly a possible construal of the VP and a fantastic result for the example at hand, it doesn't capture the full range of data. AWF is sometimes *selective* (e.g. Krifka 1991, Wold 1996). For example, as a continuation or reply to (5.27a), (5.27b) readily expresses a reading true given a party where Mary met only Bill, and Sue also met only Bill—in other words, $\{\textsf{s}\}$ is a proper subset of the people John introduced only Bill to. This reading requires the focus-sensitive *only* in (5.27b) to associate with *BILL$_F$*, and *also* to associate with *SUE$_F$* (i.e. the AWF dependencies are nested). The other conceivable pattern of selective AWF (i.e. with crossed dependencies) is grammatical, as well. As a continuation or reply to (5.28a), (5.28b) readily describes a party where Fred met only Sue, and Bill also met only Sue—in other words, $\{\textsf{b}\}$ is a proper subset of the people John introduced only to Sue. These truth conditions are distinct from those associated with (5.27b), and they require *only* to associate with *SUE$_F$* and *also* to associate with *BILL$_F$*.

(5.27) a. John only introduced BILL$_\text{F}$ to Mary.
b. He $\underbrace{\text{also only [introduced BILL}_\text{F}\text{ to SUE}_\text{F}}_{}$].

(5.28) a. John only introduced Fred to SUE$_\text{F}$.
b. He $\underbrace{\text{also only [introduced BILL}_\text{F}}_{}$ to SUE$_\text{F}$].

One possible reply to this (and a natural one!) is to suppose that AWF *is indeed* unselective, but that a covert scope shift raises one of the focused expressions out of the domain of one of the focus-sensitive operators. For (5.27b) a scope ordering giving correct truth conditions would be *also* > *SUE*$_\text{F}$ > *only* > *BILL*$_\text{F}$, and for (5.28b) an adequate scope ordering would be *also* > *BILL*$_\text{F}$ > *only* > *SUE*$_\text{F}$. I.e. in each case, the focused expression occurs within the immediate scope of the focus-sensitive operator with which it associates. In other words, the putative strategy would be to keep unselective AWF (and thus to keep the underlying Roothian architecture), but to treat apparent instances of selectivity as the result of something orthogonal to focus—namely as the result of a covert scope shift.

While this yields good results for (5.27b) and (5.28b), it isn't a fully general solution. As Wold 1996 emphasizes, association with focus across islands is potentially selective (see also Krifka 1991, 2006, Rooth 1996). For example, the second sentence of (1.21) conveys that both John and Bobby were such that we only saw the entries Hoover made about him. Such a reading requires *only* to selectively associate with *HOOVER*$_F$ (to the exclusion of *BOBBY*$_F$) and *also* to selectively associate with *BOBBY*$_F$ (to the exclusion of *HOOVER*$_F$) across a scope island boundary:

(5.29) We only saw the entries ⟨HOOVER$_\text{F}$ made about John⟩.
We also only saw the entries ⟨HOOVER$_\text{F}$ made about BOBBY$_\text{F}$⟩.
(after Rooth 1996, ex. 43)

As I noted at the outset, one of the core motivations for Rooth's 1985 semantics was the idea that AWF is insensitive to islands. Given this, it would be ad hoc to jettison this motivation for these and only these sorts of cases. Whatever resources a theory of focus alternatives and AWF uses to account for AWF's insensitivity to scope islands in simpler cases should also explain the insensitivity to scope islands in selective AWF cases such as (5.29).

5.4.3 Focus monad and composition

The AWF situation has all the hallmarks of the situation with indefinites. We have island-insensitivity, along with the possibility of selective scope-taking outside the boundaries of a scope island. This means we can directly apply what we've learned from indefinites to the case at hand. We will go in search of a monad for focus side effects and allow these side effects to take scope.

I will assume that focus is a side effect. This means focus percolation is a matter of *scope* (i.e. there is no separate focus-based mode of combination). Given this, our prediction is the following: like indefiniteness, whatever side effects are characteristic of a focus-monadic program can take exceptional scope post-evaluation. Moreover, the side effects generated by foci, like the side effects generated by indefinites, are predicted to take selective exceptional scope outside islands.

Let's see how this works. Recall that Rooth's semantics treats the interpretation function as two-dimensional, one dimension for value composition and one dimension for nondeterministic composition. We have already seen a monad for values: the Identity monad. And we have already seen a monad for nondeterminism: the Set monad. Pairing these two monads into a single monad that simultaneously does the work of value composition and nondeterministic program composition gives what I will call the Focus monad, in Definition 5.6 (see Shan 2002, who first proposed this monad for composing focus alternatives, calling it the "Pointed Powerset monad"). Focus-monadic programs are *pairs* of values and nondeterministic values: $\mathsf{M}\alpha$ is the type of a pair of type-α things and sets of type-α things. The injection function η turns a value a into a trivial program consisting of a pair of a, and a trivial Set-monadic program $\{a\}$. Sequencing in the Focus monad combines sequencing in the Identity monad with sequencing in the Set monad, training each notion of sequencing on one of the dimensions in the Focus monad (given any ordered pair m, m_0 is defined as the first element in m, and m_1 is defined as the second element in m).

Definition 5.6 (The Focus monad).

$$\begin{aligned} \mathsf{M}\alpha &::= \alpha \times (\alpha \to t) \\ a^{\eta} &:= \langle a, \{a\} \rangle \\ m_v \multimap \pi &:= \langle \pi[m_0/v]_0, \bigcup_{a \in m_1} \pi[a/v]_1 \rangle \end{aligned}$$

Monadic application in the Focus monad works analogously to Rooth, as expected. Given two pairs m and n, we do regular functional application on the first members of m and n (corresponding to Rooth's $[\![X\,Y]\!]$) and pointwise functional application (that is, NA) on the second members of m and n (corresponding to Rooth's $[\![X\,Y]\!]_f$). This justifies our claim that the Focus monad accurately captures Rooth's semantics.

Fact 5.4 (Application in the Focus monad).

$$\mathbf{A}\, m\, n = \langle \mathsf{A}\, m_0\, n_0, \{\mathsf{A}\, x\, y \mid x \in m_1 \wedge y \in n_1\} \rangle$$

Proof.

$$\begin{aligned} \mathbf{A}\, m\, n &= m_x \multimap n_y \multimap (\mathsf{A}\, x\, y)^{\eta} && \mathbf{A} \\ &= m_x \multimap n_y \multimap \langle \mathsf{A}\, x\, y, \{\mathsf{A}\, x\, y\} \rangle && \eta \\ &= m_x \multimap \langle \mathsf{A}\, x\, n_0, \{\mathsf{A}\, x\, y \mid y \in n_1\} \rangle && \multimap \\ &= \langle \mathsf{A}\, m_0\, n_0, \{\mathsf{A}\, x\, y \mid x \in m_1 \wedge y \in n_1\} \rangle && \multimap \end{aligned}$$

□

The denotations of focused constituents like *BILL$_F$* will be programs in the Focus monad; the semantics of F-marking is to map a value into a Focus-monadic program. See Definition 5.7. (One interesting difference given by the side-effects perspective is that F-marking is no longer treated syncategorematically.) As in Rooth's semantics, Alt is essentially a black box: a function from a value a into the (contextually relevant) model-theoretic alternatives to a.

Definition 5.7 (The semantics of F-marking).

$$[\![\mathrm{F}]\!]\, a := a^{\mathsf{F}} := \langle a, \mathsf{Alt}\, a \rangle$$

Meanwhile, Lift and Lower are defined as usual. The semantics of Lift, in Definition 5.8, is the semantics of the Focus monad's sequencing operator. The semantics of Lower, in Definition 5.9, means applying a tower to a trivial continuation which is the Focus monad's injection function η. See Fact 5.5 for an example of how F-marking and Lifting work to create a scopal program.

Definition 5.8 (Focus-monadic Lifting).

$$m^{\uparrow} := (\multimap)\, m$$

Definition 5.9 (Focus-monadic Lowering).

$$M^{\downarrow} := M\,\eta$$

Fact 5.5 (Lifting an F-marked node into a scope-taker).

$$\left(\begin{array}{c} \mathsf{M}e \\[1em] \text{John}_{\text{F}} \\[1em] \mathsf{j}^{\mathbf{F}} \end{array}\right)^{\uparrow} = \begin{array}{c} \dfrac{\mathsf{M}\alpha}{e} \\[1em] \text{John}_{\text{F}} \\[1em] \dfrac{\mathsf{j}^{\mathbf{F}}_{x} \multimap [\,]}{x} \end{array}$$

Let's run through some basic derivations. (To keep things as simple as possible, we will ignore cases that implicate the State.Set monad, though we'll come back to this in Section 5.4.5.) To warm up, an example derivation for *JOHN$_F$ left* is given in (5.30). Familiarly, we Lift *left* into a scope-taker, combine everything up with scopal functional application, and evaluate to give a Focus-monadic program with type $\mathsf{M}t$. The first member of the resulting pair is the expected proposition, viz. that John left. The second member of the pair is the expected alternative set containing propositions of the form x left, with x ranging over the alternatives to John.

(5.30)
$$\left(\begin{array}{cc} \dfrac{\mathsf{M}t}{e} & \dfrac{\mathsf{M}t}{e \to t} \\[1em] \text{JOHN}_{\text{F}} & \text{left} \\[1em] \dfrac{\mathsf{j}^{\mathbf{F}}_{x} \multimap [\,]}{x} & \dfrac{[\,]}{\mathsf{left}} \end{array}\right) = \begin{array}{c} \dfrac{\mathsf{M}t}{t} \\[1em] \text{JOHN}_{\text{F}}\text{ left} \\[1em] \dfrac{\mathsf{j}^{\mathbf{F}}_{x} \multimap [\,]}{\mathsf{left}\,x} \end{array}$$

$$\begin{array}{lll} \rightsquigarrow & \mathsf{j}^{\mathbf{F}}_{x} \multimap (\mathsf{left}\,x)^{\eta} & \downarrow \\ = & \langle \mathsf{j},\ \mathsf{Alt}\,\mathsf{j}\rangle_{x} \multimap (\mathsf{left}\,x)^{\eta} & \mathbf{F} \\ = & \langle \mathsf{j},\ \mathsf{Alt}\,\mathsf{j}\rangle_{x} \multimap \langle \mathsf{left}\,x,\ \{\mathsf{left}\,x\}\rangle & \eta \\ = & \underbrace{\langle \mathsf{left}\,\mathsf{j},\ \{\mathsf{left}\,x \mid x \in \mathsf{Alt}\,\mathsf{j}\}\rangle}_{\mathsf{M}t} & \multimap \end{array}$$

Now let's add a semantics of focus-sensitive VP modifiers. We will give a provisional, simple-minded semantics to *only* and *also* along the lines of Definition 5.5.[11] As before, we'll take identify the type t as the type of *sets* of worlds and ignore the distinction between presupposed and asserted entailments. Given all that, we can observe that the meanings in Definition 5.10 are in a sense simply

11 The reason the semantics is provisional: we can hear e.g. *John only met a DEMOCRAT$_F$ at the party*, as entailing that John didn't meet *any* Republicans at the party, i.e. with *only* > *a DEMOCRAT$_F$*. This reading requires VP-modifying **only** to have a semantics capable of discharging the nondeterminism generated by the indefinite. But for now, for simplicity's sake, we will leave it be.

the Focus-monadic refactoring of the meanings in Definition 5.5: *only* takes a monadic VP $\mathcal{P}$ and returns a regular VP-type meaning P such that P holds of an x at a world w iff applying the first member of $\mathcal{P}$ to x yields True at w, and there is no other property P' in the second member of $\mathcal{P}$ such that P' holds of x in w. Mutatis mutandis for *also*.

Definition 5.10 (Focus-sensitive VP-modifiers).

$$\begin{array}{cc} \mathsf{M}(e \to t) \to e \to t & \mathsf{M}(e \to t) \to e \to t \\ \\ \text{only} & \text{also} \\ \\ \textbf{only} := \lambda \mathcal{P} x w.\, \{P \mid P \in \mathcal{P}_1 \wedge P\,x\,w\} = \{\mathcal{P}_0\} & \textbf{also} := \lambda \mathcal{P} x w.\, \{P \mid P \in \mathcal{P}_1 \wedge P\,x\,w\} \supset \{\mathcal{P}_0\} \end{array}$$

Let's begin by seeing how this system derives a simple example of AWF for a sentence like *Sharon only met JOHN_F*. We Lift met into a scope-taker via an application of η and $(\multimap)$, apply $\mathbf{F}$ to j to give a $\mathsf{M}e$ and Lift the result into a scopal expression via an application of $(\multimap)$ (for both Liftings we choose a result type $\mathsf{M}(e \to t)$ to force the Focus-related side effects to take scope at the VP level), and finally combine everything via a run-of-the mill scopal application. We conclude by Lowering via an application of η to give a $\mathsf{M}(e \to t)$.

$$\left(\begin{array}{cc} \dfrac{\mathsf{M}(e \to t)}{e \to e \to t} & \dfrac{\mathsf{M}(e \to t)}{e} \\ \\ \text{met} & \text{JOHN}_{\text{F}} \\ \\ \dfrac{[\,]}{\mathsf{met}} & \dfrac{\mathsf{j}^{\mathbf{F}}_{y} \multimap [\,]}{y} \end{array}\right) = \begin{array}{c} \dfrac{\mathsf{M}(e \to t)}{e \to t} \\ \\ \text{met JOHN}_{\text{F}} \\ \\ \dfrac{\mathsf{j}^{\mathbf{F}}_{y} \multimap [\,]}{\mathsf{met}\, y} \end{array}$$

$$\rightsquigarrow \quad \mathsf{j}^{\mathbf{F}}_{y} \multimap (\mathsf{met}\, y)^{\eta} \quad \downarrow$$

All that remains is to merge *only*, type $\mathsf{M}(e \to t) \to e \to t$, with the monadic VP just derived, type $\mathsf{M}(e \to t)$. Combination is via functional application. The resulting meaning, type $e \to t$, is intuitively adequate (i.e. applying it to Sue yields the proposition that the only person Sue met was John). (The final simplification is due to the following equivalence: the second line says that the only property in the set $\{\mathsf{met}\, y \mid y \in \mathsf{Alt}\,\mathsf{j}\}$ that holds of x in w is the property of meeting John. This is the same as requiring that the only person x met in w is John.)

$$\begin{array}{rll} \textbf{only}\,(\mathsf{j}^{\mathbf{F}}_{y} \multimap (\mathsf{met}\, y)^{\eta}) & = \textbf{only}\,\langle \mathsf{met}\,\mathsf{j},\ \{\mathsf{met}\, y \mid y \in \mathsf{Alt}\,\mathsf{j}\}\rangle & \mathbf{F},\ \eta,\ \multimap \\ & = \lambda x w.\, \{P \mid P \in \{\mathsf{met}\, y \mid y \in \mathsf{Alt}\,\mathsf{j}\} \wedge P\,x\,w\} = \{\mathsf{met}\,\mathsf{j}\} & \textbf{only} \\ & = \lambda x w.\, \{y \mid \mathsf{met}\, y\, x\, w\} = \{\mathsf{j}\} & \equiv \end{array}$$

A slightly more complicated example deriving unselective association with *only* for *John only introduced BILL_F to SUE_F* works similarly. This time, we have two Focus-monadic side effects, triggered by the two focused expressions. We begin by assembling the VP *introduced BILL_F to SUE_F*, doing two instances of combination and then Lowering to give a Focus-monadic VP:

$$\left(\left(\begin{array}{c} \frac{\mathsf{M}(e\to t)}{e\to e\to e\to t} \\ \\ \text{intro'd} \\ \\ \frac{[\,]}{\mathsf{intro}} \end{array} \quad \begin{array}{c} \frac{\mathsf{M}(e\to t)}{e} \\ \\ \text{BILL}_\text{F} \\ \\ \frac{\mathsf{b}_x^{\mathsf{F}} \multimap [\,]}{x} \end{array}\right) \begin{array}{c} \frac{\mathsf{M}(e\to t)}{e} \\ \\ \text{to SUE}_\text{F} \\ \\ \frac{\mathsf{s}_y^{\mathsf{F}} \multimap [\,]}{y} \end{array}\right) = \begin{array}{c} \frac{\mathsf{M}(e\to t)}{e\to t} \\ \\ \text{intro'd BILL}_\text{F} \text{ to SUE}_\text{F} \\ \\ \frac{\mathsf{b}_x^{\mathsf{F}} \multimap \mathsf{s}_y^{\mathsf{F}} \multimap [\,]}{\mathsf{intro}\ x\ y} \end{array}$$

$$\begin{array}{rl} \rightsquigarrow & \mathsf{b}_x^{\mathsf{F}} \multimap \mathsf{s}_y^{\mathsf{F}} \multimap (\mathsf{intro}\ x\ y)^\eta \quad \downarrow \\ = & \langle \mathsf{intro\ b\ s}, \{\mathsf{intro}\ x\ y \mid x \in \mathsf{Alt\ b} \wedge y \in \mathsf{Alt\ s}\}\rangle \quad \mathbf{F}, \eta, \multimap \end{array}$$

Applying **only** to this pair gives the following property:

$$\lambda zw.\, \{P \mid P \in \{\mathsf{intro}\ x\ y \mid x \in \mathsf{Alt\ b} \wedge y \in \mathsf{Alt\ s}\} \wedge P\ z\ w\} = \{\mathsf{intro\ b\ s}\}$$

...Which is equivalent to the pair-quantification derived in the Rooth semantics:

$$\lambda zw.\, \{\langle x,\ y\rangle \mid \mathsf{intro}\ x\ y\ z\ w\} = \{\langle \mathsf{b},\ \mathsf{s}\rangle\}$$

5.4.4 Selective derivations and islands

Next, I'll give an analysis of the selective AWF reading of *John also only introduced BILL$_F$ to SUE$_F$* depicted in (5.27a), i.e. where *only* associates with *BILL$_F$*, and *also* associates with *SUE$_F$*, and then sketch how the crossed-AWF-dependencies reading is derived.

We begin by composing the VP *introduced BILL$_F$ to SUE$_F$*. We externally Lift *BILL$_F$* and internally Lift *SUE$_F$*, leaving the side effects associated the former on the second story and raising the side effects associated with *SUE$_F$* onto the third story. After composing everything up, we collapse the bottom two stories into a focus-monadic VP-type meaning, viz. $\mathsf{M}(e \to t)$:

$$\left(\left(\begin{array}{c} \frac{\frac{\mathsf{M}(e\to t)}{\mathsf{M}(e\to t)}}{e\to e\to e\to t} \\ \\ \text{intro'd} \\ \\ \frac{\frac{[\,]}{[\,]}}{\mathsf{intro}} \end{array} \quad \begin{array}{c} \frac{\frac{\mathsf{M}(e\to t)}{\mathsf{M}(e\to t)}}{e} \\ \\ \text{BILL}_\text{F} \\ \\ \frac{\frac{[\,]}{\mathsf{b}_x^{\mathsf{F}} \multimap [\,]}}{x} \end{array}\right) \begin{array}{c} \frac{\frac{\mathsf{M}(e\to t)}{\mathsf{M}(e\to t)}}{e} \\ \\ \text{to SUE}_\text{F} \\ \\ \frac{\frac{\mathsf{s}_y^{\mathsf{F}} \multimap [\,]}{[\,]}}{y} \end{array}\right) = \begin{array}{c} \frac{\frac{\mathsf{M}(e\to t)}{\mathsf{M}(e\to t)}}{e\to t} \\ \\ \text{intro'd BILL}_\text{F} \text{ to SUE}_\text{F} \\ \\ \frac{\frac{\mathsf{s}_y^{\mathsf{F}} \multimap [\,]}{\mathsf{b}_x^{\mathsf{F}} \multimap [\,]}}{\mathsf{intro}\ x\ y} \end{array}$$

$$\rightsquigarrow \quad \frac{\mathsf{s}_y^{\mathsf{F}} \multimap [\,]}{\mathsf{b}_x^{\mathsf{F}} \multimap (\mathsf{intro}\ x\ y)^\eta} \quad \downarrow$$

At this point, the bottom story of this tower is of the right type, $\mathsf{M}(e \to t)$, to combine directly with

only (type $\mathsf{M}(e \to t) \to e \to t$). We merge *only*, Lifting it into a scopal expression, and combining everything into a two-level tower. Because the Focus-monadic side effects associated with $BILL_F$ are on the bottom level of the tower, they're what *only* ends up associating with. This discharges $BILL_F$'s side effects for good; the bottom level of the resulting tower is a value with type $e \to t$. This means we can apply Lower, which yields a $\mathsf{M}(e \to t)$, with SUE_F's side effects scoping over **only**, which scopes over $BILL_F$.

$$\left(\begin{array}{cc} \dfrac{\mathsf{M}(e \to t)}{\mathsf{M}(e \to t) \to e \to t} & \dfrac{\mathsf{M}(e \to t)}{\mathsf{M}(e \to t)} \\[2ex] \text{only} & \text{intro'd BILL}_\text{F}\text{ to SUE}_\text{F} \\[2ex] \dfrac{[\,]}{\textbf{only}} & \dfrac{\mathsf{s}^{\mathbf{F}}_y \multimap [\,]}{\mathsf{b}^{\mathbf{F}}_x \multimap (\mathsf{intro}\; x\; y)^{\eta}} \end{array}\right) = \begin{array}{c} \dfrac{\mathsf{M}(e \to t)}{e \to t} \\[2ex] \text{only intro'd BILL}_\text{F}\text{ to SUE}_\text{F} \\[2ex] \dfrac{\mathsf{s}^{\mathbf{F}}_y \multimap [\,]}{\textbf{only}\;(\mathsf{b}^{\mathbf{F}}_x \multimap (\mathsf{intro}\; x\; y)^{\eta})} \end{array}$$

$$\leadsto \quad \mathsf{s}^{\mathbf{F}}_y \multimap (\textbf{only}\;(\mathsf{b}^{\mathbf{F}}_x \multimap (\mathsf{intro}\; x\; y)^{\eta}))^{\eta} \quad \downarrow$$

The final step is to merge in *also*. This has the effect of capturing the Focus-monadic side effects generated by SUE_F. We are left with a regular value, type $e \to t$, whose meaning is the intuitively correct one (this takes some calculating using the meanings given for **also** and **only** along the lines of the calculations we did for *Sue only met JOHN*$_F$; I'll leave them out for readability's sake). Notice that the resulting scope ordering, *also* > SUE_F > *only* > $BILL_F$, is exactly what was proposed earlier as a way to derive the selective AWF reading while retaining an unselective account of AWF.

$$\left(\begin{array}{cc} \mathsf{M}(e \to t) \to e \to t & \mathsf{M}(e \to t) \\[2ex] \text{also} & \text{only intro'd BILL}_\text{F}\text{ to SUE}_\text{F} \\[2ex] \textbf{also} & \mathsf{s}^{\mathbf{F}}_y \multimap (\textbf{only}\;(\mathsf{b}^{\mathbf{F}}_x \multimap (\mathsf{intro}\; x\; y)^{\eta}))^{\eta} \end{array}\right)$$

$$\begin{aligned} &= \quad \textbf{also}\;(\mathsf{s}^{\mathbf{F}}_y \multimap (\textbf{only}\;(\mathsf{b}^{\mathbf{F}}_x \multimap (\mathsf{intro}\; x\; y)^{\eta}))^{\eta}) \quad \mathsf{A} \\ &= \quad \lambda z w.\, \{y \mid \{x \mid \mathsf{intro}\; x\; y\; z\; w\} = \{\mathsf{b}\}\} \supset \{\mathsf{s}\} \quad \textbf{also}, \textbf{only} \end{aligned}$$

Of course, could have gotten this with QR anyway, as with the putative scope-shift account we mooted for the Roothian theory. However, we make a prediction that theory does not: namely, that *selective* AWF out of islands is possible. As a first step, notice that since scope is what's responsible for all side effect percolation, and residual side effects are free to take scope post-evaluation, we predict that Focus-monadic side effects will be able to take scope post-Reset:

$$\begin{aligned}\left(\frac{\mathsf{j}^{\mathbf{F}}_x \multimap [\,]}{\mathsf{left}\,x}\right)^{\downarrow\uparrow} &= (\mathsf{j}^{\mathbf{F}}_x \multimap (\mathsf{left}\,x)^{\eta})^{\uparrow} && \downarrow\\ &= \frac{(\mathsf{j}^{\mathbf{F}}_x \multimap (\mathsf{left}\,x)^{\eta})_p \multimap [\,]}{p} && \uparrow\\ &= \frac{\mathsf{j}^{\mathbf{F}}_x \multimap (\mathsf{left}\,x)^{\eta}_p \multimap [\,]}{p} && \text{Assoc}\\ &= \frac{\mathsf{j}^{\mathbf{F}}_x \multimap [\,]}{\mathsf{left}\,x} && \text{LeftID}\end{aligned}$$

Thus, even though we use scope shift to manage the percolation of focus alternatives, we readily predict that side effects triggered by focus will freely scope out of scope islands—i.e. out of any domain that is obligatorily evaluated.

Finally, higher-order Focus-monadic programs allows us to explain selective AWF out of islands, as in (5.29). Recall how selective exceptional scope with multiple indefinites worked in Chapter 4. We first construct a higher-order, fully evaluated State.Set program with type $\mathsf{MM}t$, relying on the polymorphism inherent in the State.Set monad's Lift operation. This object preserves enough structure that it can be unfolded post-evaluation into a three-level tower; post-Reset, both indefinites can take exceptional scope as they pleased, potentially separately from each other. The same thing works here. Because $(\multimap)$ is polymorphic, we predict that fully evaluated $\mathsf{MM}t$ objects are possible, which retain enough structure to be recursively unfolded post-evaluation:

$$\begin{array}{c}\dfrac{\mathsf{MM}t}{\dfrac{\mathsf{M}t}{t}}\\[1ex] \text{BILL}_{\text{F}}\text{ met SUE}_{\text{F}}\\[1ex] \dfrac{\mathsf{s}^{\mathbf{F}}_x \multimap [\,]}{\dfrac{\mathsf{b}^{\mathbf{F}}_y \multimap [\,]}{\mathsf{met}\,x\,y}}\end{array} \rightsquigarrow \underbrace{\mathsf{s}^{\mathbf{F}}_x \multimap (\mathsf{b}^{\mathbf{F}}_y \multimap (\mathsf{met}\,x\,y)^{\eta})^{\eta}}_{\mathsf{MM}t} \quad \downarrow, \downarrow$$

Thus, selective AWF is possible even across islands: Focus side effects in a single domain of evaluation can occupy different levels post-Reset and are free to take differential scope. Just as the nondeterministic side effects associated with indefinites can take selective, articulated scope out of islands, so can Focus-monadic side effects, via the same sort of mechanism. It is novel to formally identify the mechanisms by which indefinites (potentially selectively) scope out of islands and by which focus alternatives (potentially selectively) scope out of islands.

An additional case is of interest. Krifka 1991 points out that selective AWF is possible in cases where two focus-sensitive adverbs seem to associate with a single word. The second sentence in

(5.31) can mean that John only drank water on some occasion, and there was some other occasion that he only drank something other than water.

(5.31) John once only drank WINE$_{\mathrm{F}}$.
John also once only drank [WATER$_{\mathrm{F}}$]$_{\mathrm{F}}$.

Our semantics extends to cases like this straightaway. As the above structure suggests, I assume (with Krifka) that *WATER* in the second sentence is doubly-focused. We combine *WATER* with its first F-marking, Lift, and then combine it with its second F-marking, as follows:

(5.32)
$$\left(\begin{array}{cc} \frac{\mathsf{M}(e \to t)}{e} & \frac{\mathsf{M}(e \to t)}{e \to \mathsf{M}e} \\ \mathrm{WATER_F} & \mathrm{F} \\ \frac{\mathsf{water}^{\mathbf{F}}_x \multimap [\,]}{x} & \frac{[\,]}{\lambda a.\, a^{\mathbf{F}}} \end{array}\right) = \begin{array}{c} \frac{\mathsf{M}(e \to t)}{\mathsf{M}e} \\ [\mathrm{WATER_F}]_{\mathrm{F}} \\ \frac{\mathsf{water}^{\mathbf{F}}_x \multimap [\,]}{x^{\mathbf{F}}} \end{array}$$

At this point, we can Lift the bottom story to give a three-story tower parallel to the three-level tower derived for *introduced BILL$_F$ to SUE$_F$*, i.e. with Focus-monadic side effects on two levels. The rest of the derivation is just adding in *drank*, Lowering and combining with *only*, and finally Lowering and combining with *also*. The result is entirely parallel to the previous selective AWF case, but this time **only** associates with $x^{\mathbf{F}}$:

$$\textbf{also}\,(\mathsf{water}^{\mathbf{F}}_x \multimap (\textbf{only}\,(x^{\mathbf{F}}_y \multimap (\mathsf{drank}\,y)^{\eta}))^{\eta})$$

The result is equivalent to the following property, which holds of a z in a w iff the set of things z only drank in w includes water and something besides water (as Wold 1996 notes, this truth condition is technically contradictory since we are ignoring the quantification over events/times introduced by *once* in order to keep things simple):

$$\lambda zw.\,\{x \mid \{y \mid \mathsf{drank}\,y\,z\,w\} = \{x\}\} \supset \{\mathsf{water}\}$$

Unlike the previous case, It is impossible here to derive the reverse pattern of of selective AWF, i.e. where the relative scopes of $\mathsf{water}^{\mathbf{F}}$ and $x^{\mathbf{F}}$ are reversed: unlike the previous case, here the latter depends on the former for its value.

5.4.5 Multiple underlying monads

We've been ignoring the State.Set monad so far. In this section I will briefly examine what happens when we have *two* underlying monads running around in the grammar. We'll distinguish the State.Set monad with a subscripted "1" and the Focus monad with a subscripted "2".

Perhaps surprisingly, the underlying monads $\mathcal{M}_1$ and $\mathcal{M}_2$ play well together due to their inherent polymorphism. Consider the derivation of *a linguist met POLLY$_F$* below. We construct an $\mathsf{M}_2\mathsf{M}_1 t$

program by building a three-level tower where Focus-monadic effects live on the top level, and State.Set-monadic effects on the second. The result after evaluating is a layered $\mathsf{M}_2\mathsf{M}_1 t$ program which incurs both State.Set and Focus side effects.

$$\left(\begin{array}{cc}
\begin{array}{c} \mathsf{M}_2\mathsf{M}_1 t \\ \hline \mathsf{M}_1 t \\ \hline e \end{array} & \begin{array}{c} \mathsf{M}_2\mathsf{M}_1 t \\ \hline \mathsf{M}_1 t \\ \hline e \to t \end{array} \\[2ex]
\text{a linguist} & \text{met POLLY}_{\text{F}} \\[2ex]
\begin{array}{c} [\,] \\ \hline \textbf{a.ling}_x \multimap_1 [\,] \\ \hline x \end{array} & \begin{array}{c} \mathsf{p}^{\mathbf{F}}_y \multimap_2 [\,] \\ \hline [\,] \\ \hline \mathsf{met}\ y \end{array}
\end{array}\right) = \begin{array}{c}
\begin{array}{c} \mathsf{M}_2\mathsf{M}_1 t \\ \hline \mathsf{M}_1 t \\ \hline t \end{array} \\[2ex]
\text{a linguist met POLLY}_{\text{F}} \\[2ex]
\begin{array}{c} \mathsf{p}^{\mathbf{F}}_y \multimap_2 [\,] \\ \hline \textbf{a.ling}_x \multimap_1 [\,] \\ \hline \mathsf{met}\ y\ x \end{array}
\end{array}$$

$$\begin{array}{rcl}
\rightsquigarrow & \mathsf{p}^{\mathbf{F}}_y \multimap_2 (\textbf{a.ling}_x \multimap_1 (\mathsf{met}\ y\ x)^{\eta_1})^{\eta_2} & \downarrow_1, \downarrow_2 \\
= & \underbrace{\langle \textbf{a.ling}_x \multimap_1 (\mathsf{met}\ \mathsf{p}\ x)^{\eta_1}, \{\textbf{a.ling}_x \multimap_1 (\mathsf{met}\ y\ x)^{\eta_1} \mid y \in \mathsf{Alt}\ \mathsf{p}\}\rangle}_{\mathsf{M}_2\mathsf{M}_1 t} & \mathbf{F}, \multimap_2
\end{array}$$

Notice also that we can recursively unfold the evaluated program back into a three-tiered tower to complete a Reset. Thus the Focus- and State.Set- monadic side effects in this multi-monadic grammar are characterized by an immunity to Reset and the possibility of higher-order structure.

The reverse layering of the two underlying monads, viz. $\mathsf{M}_1\mathsf{M}_2\alpha$, is possible as well. The example below exploits this to give a *pronoun* scope over an $\mathbf{F}$. The upshot is that alternatives introduced by $\mathbf{F}$ will depend on *how the pronoun is resolved*:

$$\begin{array}{c}
\begin{array}{c} \mathsf{M}_1\mathsf{M}_2 t \\ \hline \mathsf{M}_2 t \\ \hline e \end{array} \\[2ex]
\text{HE}_{\text{F}} \\[2ex]
\begin{array}{c} \textbf{pro}_x \multimap_1 [\,] \\ \hline x^{\mathbf{F}}_y \multimap_2 [\,] \\ \hline y \end{array}
\end{array}$$

There is independent motivation for focus sets whose form depends on how a pronoun is resolved—i.e. for the possibility of programs of type $\mathsf{M}_1\mathsf{M}_2 t$. Jacobson 2000b, 2004 argues that resolution-dependent focus sets are an essential to explaining *focused bound pronouns*, example (5.33), and essential to explaining a constraint on antecedent-contained deletion known as Kennedy's puzzle, example (5.34) (e.g. Kennedy 1994, Heim 1997, Kennedy 2014; see Sauerland 2004 for further discussion of both focused bound pronouns and Kennedy's puzzle, along with an alternative account, Mayr 2012 for a detailed discussion of focused bound pronouns, and Charlow 2014 for further exploration of the Focus monad's predictions with respect to antecedent-contained deletion).

Finally, cases like (5.35a) and (5.35b) seem to require that the form of an alternative set can vary with the way a pronoun is resolved. In (5.35a), if e.g. John has three people in his nuclear family and Bill has five, we want to negate two non-mother-implicating propositions for John and four non-mother-implicating propositions for Bill. In (5.35b), we intuitively want the alternatives to vary per choice of boy; the second sentence should entail that for each boy x, x thinks x's mom is smart (in addition to thinking x is smart).[12]

(5.33) Every third-grade boy$_i$ likes his$_i$ mom, and every FOURTH$_\text{F}$ grade boy$_j$ likes HIS$_{j,\text{F}}$ mom.

(5.34) *Polly visited every town in every country ERIC$_\text{F}$ did ~~visit~~.
Kennedy 1994, ex. 3b

(5.35) a. As far as family members, every boy here$_i$ only respects [his$_i$ MOTHER]$_\text{F}$.
b. Every boy$_i$ thinks his$_i$ mom's smart. Every boy$_i$ also thinks HE$_{i,\text{F}}$'s smart.

In sum, having two underlying monads, State.Set and Focus, poses no problems for interpretation. It is possible to derive programs with type $\mathsf{M}_2\mathsf{M}_1\alpha$, as well as programs with type $\mathsf{M}_1\mathsf{M}_2\alpha$. And despite needing to keep the books for what seems like a fairly complex side effects regime (we have State for anaphora and updating the state, Set for nondeterminism, and Focus on top of everything to handle focus alternatives), *nothing about the way things combine has changed*! Composition is still uniformly handled by **S**. (In general, continuations offer a way to "combine" two monads without directly defining a combined monad: simply let the monads take scope on different levels.) More work needs to be done to examine the predictions of this theory (and, more generally, of the possibility for multiple underlying monads), but the initial state of play seems promising.

5.5 A compositional alternative semantics for indefinites

The moral so far is that *whatever* side effects a theorist sees evidence for, the inevitable prediction is that they'll be the sorts of things that survive evaluation and take scope out of scope islands. To underline this point, we will conclude with a bit of alternate history. I consider what happens when we swap out the underlying State.Set monad (*state-changing* plus nondeterministic side effects) for an underlying Reader.Set monad (*state-sensitive* plus nondeterministic side effects). In particular, I argue that doing so has benefits for semanticists who wish to exploit a nondeterministic semantics for exceptional scope, but who do not wish to take on the dynamic account of anaphora.

Alternative semantics treats indefinites and/or questions and/or disjunctions as nondeterministic, using a nondeterministic perspective on meaning and meaning composition to relate sentences with sets of meanings. Yet a robust compositional formulation of alternative semantics has been difficult to pin down. Kratzer & Shimoyama's 2002 treatment has been criticized by Shan 2004 for its treatment of predicate abstraction and binding (details to follow). While a solution to Shan's criticisms has been

12 Rooth's 1985 official semantics does not allow for alternatives whose form depends on how pronouns are resolved. Wold's 1996 account of focused expressions as variables bound by focus-sensitive adverbs, though it does not rely on nondeterministic modes of composition, also does not allow for the relevant covarying interpretations of constructions such as (5.35a) and (5.35b) where a focused expression is bound into.

proposed by Novel & Romero 2010, I argue that this solution does not extend to an alternatives-based account of indefiniteness.

This section sketches a fully compositional treatment of alternative semantics. In so doing, I illustrate how the explanation covered in Chapter 4 should be taken to be modular. While I think it's salutary to handle cross-sentential/donkey anaphora and exceptional scope simultaneously (see for instance the difficulties mentioned for E-type accounts of sloppy donkey pronouns in Section 5.2.2), the account of exceptional scope and nondeterministic percolation can be sensibly cleaved off from the account of anaphora. While the unified account of cross-sentential/donkey anaphora and exceptional scope detailed in Chapter 4 relies on an underlying State.Set monad, a simple tweak—from the StateT monad transformer to the ReaderT monad transformer, and thus, to the Reader.Set monad—gives us a compositional semantics whose results are directly in the alternative-semantic mold. This offers a clear reckoning of the choice points in the construction of a grammar, while allowing semanticists who prefer static accounts of cross-sentential/donkey anaphora to nevertheless use my account of exceptional scope.

5.5.1 Alternative semantics

Alternatives-based theories of indefinites and disjunction (e.g. Zimmermann 2000, Kratzer & Shimoyama 2002, Shimoyama 2006, Alonso-Ovalle 2006, Aloni 2007, Mascarenhas 2009, Groenendijk & Roelofsen 2009, Roelofsen 2013) use nondeterminism to explain various properties of indefinites, for example their exceptional scope ability (Kratzer & Shimoyama 2002, Shimoyama 2006). This section sketches an alternatives-based semantics for indefinites and disjunction that I call 'AltS', hewing closest to Kratzer & Shimoyama 2002, Shimoyama 2006, Alonso-Ovalle 2006.[13] AltS offers a way to give indefinites and disjunctions wide scope without resorting to ad hoc, island-violating scoping mechanisms, and manages this via a nondeterministic perspective on meaning.

In AltS constituents denote functions from a state s into nondeterministic sets of meanings. Some of these meanings are deterministic—i.e. they inevitably return singletons—but some—e.g. the meanings of indefinites and disjunctions, see Definition 5.11—are nondeterministic and potentially output more than one value.

Definition 5.11 (AltS indefinites and disjunctions).

$$[\![\text{a linguist}]\!] \coloneqq \textbf{a.ling} \coloneqq \lambda s.\, \{x \mid \mathsf{ling}\, x\}$$
$$[\![X \text{ or } Y]\!] \coloneqq \lambda s.\, [\![X]\!]\, s \cup [\![Y]\!]\, s$$

This pervasive shift in types mandates a new, alternative-friendly composition rule which performs state-sensitive nondeterministic application (SSNA)—i.e. the mode of application characterized by the Reader.Set monad way back in Chapter 2 (see immediately below for a reminder). One upshot of

13 AltS should not be taken to reflect Kratzer & Shimoyama's views about English indefinites (let alone disjunctions). Kratzer & Shimoyama are concerned with giving a semantics for *indeterminate pronouns* in Japanese and German and make no claims that their analysis is the right one for English indefinites (though they express optimism that their theory could underwrite a general theory of indefinites and indefiniteness cross-linguistically).

SSNA in AltS is that constituents inherit the nondeterminism of their parts. A couple examples of using SSNA to interpret sentences are given in Fact 5.6.

$$\begin{aligned}\mathsf{SSNA}\, m\, n \coloneqq \lambda s.\, \{\, \mathsf{A}\, x\, y \mid x \in m\, s \,\wedge \\ y \in n\, s \,\}\end{aligned}$$

Fact 5.6 (AltS sentence meanings).

$$[\![\text{a linguist left}]\!] = \lambda s.\, \{\mathsf{left}\, x \mid \mathsf{ling}\, x\}$$
$$[\![\text{Bill or John left}]\!] = \lambda s.\, \{\mathsf{left}\, x \mid x \in \{\mathsf{b}, \mathsf{j}\}\}$$

Nondeterminism takes widest scope by default in AltS. AltS can tame nondeterminism via a syncategorematic rule for interpretation that has the effect of existential closure over outputs. See Definition 5.12. This rule flattens a set of alternatives into a deterministic singleton set whose lone member claims that there is a true element in $[\![X]\!]\, s$. For instance, $[\![\exists\ [\text{a linguist left}]]\!] = \lambda s.\, \{\exists x.\, \mathsf{ling}\, x \wedge \mathsf{left}\, x\}$. Notice it's defined syncategorematically.

Definition 5.12 (AltS existential closure).

$$[\![\exists\, X]\!] \coloneqq \lambda s.\, \{\exists p.\, p \in [\![X]\!]\, s \wedge p\}$$

Initially, the state of play in AltS seems promising vis à vis exceptional scope-taking: we have a way to percolate existential scope (qua nondeterminism) over operators and out of islands (namely, alternative-friendly functional application coupled with a nondeterministic semantics for indefinites and disjunctions), as in (5.36a), and a device for *halting* the upward trajectory of nondeterminism at arbitrary points in a tree (namely, existential closure over outputs), as in (5.36b) (I assume $[\![\text{not}]\!] \coloneqq \lambda s.\, \{\lambda p.\, \neg p\}$). The effect of percolating nondeterminism around a negation, as in (5.36a), is to point-wise *flip the polarity* of every member of $[\![\text{a linguist left}]\!] = \lambda s.\, \{\mathsf{left}\, x \mid \mathsf{ling}\, x\}$, with the effect that $[\![\text{it isn't the case that a linguist left}]\!] = \lambda s.\, \{\neg\mathsf{left}\, x \mid \mathsf{ling}\, x\}$.

(5.36a) $[\![\exists\ [\text{it isn't the case that a linguist left}]]\!] = \lambda s.\, \{\exists x.\, \mathsf{ling}\, x \wedge \neg\mathsf{left}\, x\}$

(5.36b) $[\![\text{it isn't the case that } \exists\ [\text{a linguist left}]]\!] = \lambda s.\, \{\neg\exists x.\, \mathsf{ling}\, x \wedge \mathsf{left}\, x\}$

Something similar holds for the alternatives created by disjunctions. By default, these alternatives will percolate up the spine of the tree, up until the nearest existential closure operator. Thus, it is predicted that disjunctions have a way to scope out of islands.

5.5.2 Issues: selectivity and binding

But using AltS for exceptional scope has some problems. To begin, AltS doesn't offer enough structure when two nondeterministic expressions occur in a single constituent. Both (5.37) and (5.38) have but a single layer of nondeterminism. Thus, for any operator outside e.g. (5.37), the only option is to give both indefinites scope over the operator, or to give them both scope under the operator (I gave evidence against this prediction in Section 4.5.2). The problem recurs in (5.38): there is no way to give the disjunction scope over some operator without giving both of the indefinites scope over the operator, as well. We have also seen that this is problematic: a disjunction can take (exceptional) scope over some operator without either disjunct taking scope over that operator.

(5.37) $[\![\text{Bill thinks} \langle\text{a linguist met a dean}\rangle]\!] = \lambda s.\, \{\mathsf{thinks}\,(\mathsf{met}\,y\,x)\,\mathsf{b} \mid \mathsf{ling}\,x \wedge \mathsf{dean}\,y\}$

(5.38) $[\![\text{every child ate a steak or a burger}]\!] = \lambda s.\, \{\forall x.\, \mathsf{child}\,x \Rightarrow \mathsf{ate}\,y\,x \mid \mathsf{steak}\,y \vee \mathsf{burger}\,y\}$

Binding also presents issues. First, notice that AltS makes no provisions for cross-sentential or donkey anaphora: propositions output truth values rather than stacks or assignments, and so there is no obvious way to leverage DyS to provide an account of how an indefinite binds pronouns outside its scope. Of course, AltS isn't committed to a dynamic theory of donkey anaphora. But the issues with binding in AltS run deeper than this. As Shan 2004 points out, Kratzer & Shimoyama 2002 propose a rule for predicate abstraction in order to bind quantifier traces and pronouns that does not work as it should (recall from Section 2.3.4 that we could not define a version of the Reader monad's Derived VP rule in the Reader.Set monad). See Definition 5.13 ('$s[x/n]$' designates the s' differing from s at most in that $s'_n = x$).[14]

Definition 5.13 (AltS-style predicate abstraction).

$$[\![\lambda_n\, X]\!] \coloneqq \lambda s.\, \{f \mid \forall a.\, f\,a \in [\![X]\!]\, s[a/n]\}$$

Though Definition 5.13 delivers something with the correct type, the meaning that results does not manage to inherit the nondeterminism in $[\![X]\!]$ in a tenable way. The problem is that there are *too many* functions in the set returned. Intuitively, for any assignment s, $[\![\lambda_n\, X]\!]\, s$ should have the same cardinality as $[\![X]\!]\, s$. Predicate abstraction should neither create nor eliminate nondeterminism. But this will not in general be the case for Definition 5.13: there are many models, constituents X, integers n, and assignments s for which $|[\![\lambda_n\, X]\!]\, s| > |[\![X]\!]\, s|$!

An example where we attempt to abstract over a QR trace illustrates an empirical difficulty with this setup. Imagine a model with three people $\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, and three papers $\{\mathsf{p_a}, \mathsf{p_b}, \mathsf{p_c}\}$, such that everyone read *exactly two* papers, as follows: a didn't read $\mathsf{p_a}$, b didn't read $\mathsf{p_b}$, and c didn't read $\mathsf{p_c}$. *Nobody read a paper* doesn't have a reading that's true in this scenario. It has only a wide-scope-indefinite reading on which there's a single paper nobody read (false since no paper was neglected) and a narrow-scope-indefinite reading on which nobody read any papers (false since everyone did some paper-reading). Yet Definition 5.13 predicts that *nobody read a paper* has a true, quasi-functional reading here. A possible LF and corresponding interpretation are given in (5.39a) (I'm assuming here that $[\![\text{nobody}]\!] \coloneqq \lambda s.\, \{\lambda k.\, \neg\exists x.\, k\,x\}$). Now, one possibility for f in (5.39a) is the function oops, defined in (5.39b). It's straightforward to check that oops jibes with Definition 5.13: for every person x, $\mathsf{oops}\,x \in \{\mathsf{read}\,y\,x \mid \mathsf{paper}\,y\}$. Yet, since no x read p_x, $\neg\exists x.\, \mathsf{oops}\,x$, and so *nobody read a paper* is incorrectly predicted true in this scenario (more precisely, the sentence's

14 Alonso-Ovalle 2006: 116 also adopts this rule for functional abstraction, with similarly problematic results.

truth conditions are that nobody read *every* paper).[15]

(5.39a) $[\![\text{nobody } [\lambda_1\ [t_1 \text{ read a paper}]]]\!] = \lambda s.\,\{\neg\exists x.\, f\,x \mid \forall x.\, f\,x \in \{\mathsf{read}\ y\ x \mid \mathsf{paper}\ y\}\}$

(5.39b) $$\mathsf{oops} := \begin{cases} \mathsf{a} & \mapsto\ \mathsf{read}\ \mathsf{p_a}\ \mathsf{a} \\ \mathsf{b} & \mapsto\ \mathsf{read}\ \mathsf{p_b}\ \mathsf{b} \\ \mathsf{c} & \mapsto\ \mathsf{read}\ \mathsf{p_c}\ \mathsf{c} \end{cases}$$

In a nutshell, (5.13) over-generates functional alternatives. In addition to the correct "uniform" functions—i.e. for (5.39a) those f that implicate the same paper p for each x—it inevitably outputs "mixed" functions like (5.39b). Worse, as Shan 2004 emphasizes, there *is* no way to move from a function into sets of propositions into a set of properties while guaranteeing the functions are uniform as desired.

The problem goes beyond using QR for scope shift. Since AltS relies on AltS-style abstraction for binding, the issue recurs whenever a quantifier binds into a constituent containing an indefinite such as *nobody$_i$ told her$_i$ student about a famous linguist*. In addition, the problem is not just one of over-generation: we *need* to generate wide-scope interpretations for indefinites, including in cases when an indefinite is separated by a scope island from a position over operators like *nobody* or *everybody*. But AltS-style abstraction will in general generate truth conditions too weak to underlie such readings. Consider a similar case but where the operator to be out-scoped is upward-monotone (e.g. *everybody*); AltS-style abstraction fails to generate a wide-scope reading for the indefinite. The truth conditions of *everybody claimed he met a famous linguist* generated below are (fudging on the state-sensitivity of the complement of *claim*) identical to those of the narrow-scope reading; they allow famous linguists to vary with claimers.

$$[\![\text{everybody } [\lambda_1\ [t_1 \text{ claimed } \langle \text{he}_1 \text{ met a famous linguist}\rangle]]]\!] = \\ \lambda s.\,\{\forall x.\, f\,x \mid \forall x.\, f\,x \in \{\mathsf{claimed}\ (\mathsf{met}\ y\ x)\ x \mid \mathsf{famous.ling}\ y\}\}$$

Novel & Romero 2010 argue that these difficulties are surmountable, proposing to re-conceive AltS meanings as sets of state-sensitive objects, e.g. such that $[\![t_1 \text{ read a paper}]\!] = \{\lambda s.\, \mathsf{read}\ y\ s_1 \mid \mathsf{paper}\ y\}$. This entails rejiggering application as in Definition 5.14, i.e. nondeterministically performing SSA on $[\![X]\!]$ and $[\![Y]\!]$.[16] This fixes the problem; a well-behaved abstraction operation can be defined, as in Definition 5.15 (where $m^{\lhd n}$ is the assignment-function parallel of our Derived VP rule for the Reader monad in Section 2.3.2).

15 This is reminiscent of Schwarz's 2001's criticisms of choice-functional accounts of the scope of indefinites discussed in Chapter 4 and recapitulates issues for *Skolemized* choice functions discussed by Chierchia 2001, Schwarz 2001.

16 As mentioned in Chapter 2's fn. 3, NSSA is the compositional rule assumed by Rooth 1985. Intriguingly, there *is* no monad for programs that are sets of state-sensitive objects, and thus, no monad whose monadic application rule is NSSA. A SetT monad transformer does exist, but applying it to the Reader monad gives back... the Reader.Set monad! Thanks to Dylan Bumford for discussing this point with me.

Definition 5.14 (Nondeterministic state-sensitive functional application).

$$[\![X\,Y]\!] \coloneqq \mathsf{NSSA}\,[\![X]\!]\,[\![Y]\!] \coloneqq \{\lambda s.\,\mathsf{A}\,(m\,s)\,(n\,s) \mid m \in [\![X]\!] \land n \in [\![Y]\!]\,\}$$

Definition 5.15 (NSSA abstraction).

$$[\![\lambda_n\,X]\!] \coloneqq \{m^{\triangleleft_n} \mid m \in [\![X]\!]\}$$
$$m^{\triangleleft_n} \coloneqq \lambda sa.\,m\,s[a/n]\,a$$

But though NSSA solves the problem with predicate abstraction, it is not clear how to give an NSSA-style analysis to cases where an indefinite is bound into, for example Schwarz 2001's *no candidate$_i$ submitted a paper he$_i$ wrote*. There's no obvious denotation for *a paper he$_i$ wrote* in terms of sets of state-sensitive objects. How many members should such a set have? The answer, it seems, should depend on how *he$_i$* is resolved, which militates in favor of a $\lambda s.\,\{\cdots\}$ layering (again, this issue is presaged in Shan's 2004 discussion of constituent questions involving wh-phrases such as *which of his paintings*). I believe any conceivable way of addressing this point (including the one mooted by Novel & Romero) runs afoul of the BRC: since nondeterminism out-scopes assignment sensitivity by default in NSSA, nothing can prevent the nondeterminism engendered by *a paper he$_i$ wrote* from percolating around *nobody$_i$* and erroneously deriving a wide-scope interpretation for the indefinite. To take an example, Orin Percus (p.c.) suggests $\{\lambda s.\,f \mid f \in \mathsf{CF}\}$ as a possible NSSA denotation for $[\![\text{a}]\!]$ (with CF the set of choice functions). This makes quick work of the how-many-members question for *a paper he$_i$ wrote* (i.e. potentially as many as there are functions in the type $s \to e$), but the price is erroneously predicting BRC violations along the lines explored in 4.7.1:

$$[\![\text{nobody }[\lambda_1\,[t_1\text{ submitted a paper he}_1\text{ wrote}]]]\!] =$$
$$\{\lambda s.\,\neg\exists x.\,\mathsf{submitted}\,(f\,(\lambda y.\,\mathsf{paper}\,y \land \mathsf{wrote}\,y\,x))\,x \mid f \in \mathsf{CF}\}$$

Absent additional restrictions on the sorts of choice functions which make it into CF, there is (a constant function into) a True boolean in this set iff nobody submitted *every* paper they wrote. The truth conditions reduce to those of the problematic BRC-disrespecting choice-functional derivation.

5.5.3 Back to the Reader.Set monad

I've identified two main issues with AltS. One: it lacks the structure to deal with selective exceptional scope-taking. Two: it lacks a robust account of binding and anaphora. What's the diagnosis for what went wrong here? In the first case, unstructured alternative sets are simply baked into the presuppositions of the semantics: if we expect to end up with sets of simple values (and design our semantics so as to secure this result), we will surely end up with sets of values. In the second case, the root of the problem is using *pseudo-scope* mechanisms—i.e. SSNA or NSSA—to model the scope-taking of indefinites and indefiniteness, even as their scope-taking shares so many of the hallmarks of bona fide scope-taking (e.g. respecting the BRC).

Solving these problems does not require the full power of the State.Set monad. If one does not wish to go whole-hog to give a unified explanation of cross-sentential/donkey anaphora and exceptional scope in one fell swoop, but still wishes to say something about the exceptional scope properties of indefiniteness, my framework can respect their wishes. In other words, my theory is modular. We can easily swap out the State monad in favor of the Reader monad properties if we only wish to have a grammar that traffics in SSNA-style meanings.

The first step in saving AltS is jettisoning it, in favor of the perspective on composition that we developed in Chapter 3. As there, we will assume that there is scopal application and an underlying monad. Unlike there, in lieu of setting the underlying monad to the State.Set monad, we will set it to the Reader.Set monad—which, recalling Chapter 2, differs from the State.Set monad in that (i) it's derived by applying ReaderT to the Set monad in lieu of StateT, and (ii) not unrelatedly, it's built for reading in states rather than modifying and storing them for later. Here is a refresher on the Reader.Set monad:

$$\mathsf{M}\alpha ::= s \to \alpha \to t$$

$$a^{\eta} := \lambda s.\, \{a\}$$

$$m_v \multimap \pi := \lambda s. \bigcup_{a \in m\, s} \pi[a/v]\, s$$

Meanwhile, the apparatus for side effects composition will be exactly as it's been since Chapter 3: η and $\multimap$ are freely available type-shifters (and scopal composition is performed by $\mathbf{S}$):

Definition 5.16 (Reader.Set Lifting).

$$m^{\uparrow} := (\multimap)\, m$$

Definition 5.17 (Reader.Set Lowering).

$$m^{\downarrow} := m\, \eta$$

Likewise, here is a refresher on the Reader.Set meanings for indefinites and pronouns. Both are precisely as they were in the discussion in Chapter 2. The indefinite denotes a constant function from a state (i.e. a stack) into a set of individuals, and the pronoun denotes a non-constant function from an stack into a singleton set containing a stack-dependent value:

$$\textbf{a.ling} := \lambda s.\, \{x \mid \mathsf{ling}\, x\}$$

$$\textbf{pro} := \lambda s.\, \{s_{\top}\}$$

That is all there is to it. All we've done is swap out our lexical entries in favor of AltS-style ones and replace the underlying State.Set monad with the Reader.Set monad. Let's see what this buys us. First, we can give an analysis of exceptional scope-taking in exactly analogous terms to the explanation we gave in Chapter 4. We suppose that scope islands are domains that must be evaluated. In Fact 5.7 we Reset a scope island, namely the sentence *a linguist left*. After Lowering and Lifting, we see that the indefinite's side effects have survived and are free to keep taking scope.

Fact 5.7 (Resetting *a linguist left*).

$$\begin{aligned}
\left(\frac{\textsf{\textbf{a.ling}}_x \multimap [\,]}{\textsf{left}\,x}\right)^{\downarrow\uparrow} &= (\textsf{\textbf{a.ling}}_x \multimap (\textsf{left}\,x)^{\eta})^{\uparrow} && \downarrow \\
&= \frac{(\textsf{\textbf{a.ling}}_x \multimap (\textsf{left}\,x)^{\eta})_p \multimap [\,]}{p} && \uparrow \\
&= \frac{\textsf{\textbf{a.ling}}_x \multimap (\textsf{left}\,x)^{\eta}_p \multimap [\,]}{p} && \text{Assoc} \\
&= \frac{\textsf{\textbf{a.ling}}_x \multimap [\,]}{\textsf{left}\,x} && \text{LeftID}
\end{aligned}$$

If we suppose that quantifiers such as *every linguist* are deterministic—i.e. that they always return a single value (cf. the discussion of generalized quantifiers in Kratzer & Shimoyama 2002: 8)—we predict that the scope of quantifiers is bounded by scope islands. Along these lines, a Reader.Set semantics for a quantificational DP, namely *every linguist*, is given in Definition 5.19. As in the State.Set monad, this semantics is defined in terms of the meaning of negation, given in Definition 5.18. In both cases, the meanings differ from their State.Set counterparts only in that here we return nondeterministic values rather than nondeterministic value-stack pairs.

Definition 5.18 (Reader.Set negation).

$$\textsf{M}t \to \textsf{M}t$$

$$\text{not}$$

$$\textsf{\textbf{not}} := \lambda ms.\, \{\neg\exists a.\, a \in m\, s \wedge a\}$$

Definition 5.19 (Reader.Set universal quantifier).

$$\frac{\textsf{M}t}{e}$$

$$\text{every linguist}$$

$$\textsf{\textbf{ev.ling}} := \frac{\textsf{\textbf{not}}\,(\textsf{\textbf{a.ling}}_x \multimap \textsf{\textbf{not}}\,[\,])}{x}$$

Consider now what happens when we evaluate *every linguist left*. The result is a trivial program, and Lifting it back into a scope-taker via an application of ↑ does nothing to restore the universal's quantificational force. Evaluation forces quantification to happen, and, much as in the State.Set monad, that is that for the universal quantifier. Thus, indefinites but not universals can take scope out of scope islands—that is, beyond any domain in which evaluation is obligatory.

Fact 5.8 (Resetting *every linguist left*).

$$\begin{aligned}
\left(\frac{\textbf{ev.ling}\,(\lambda x.\,[\,])}{\mathsf{left}\,x}\right)^{\downarrow\uparrow} &= (\textbf{ev.ling}\,(\lambda x.\,(\mathsf{left}\,x)^{\eta}))^{\uparrow} && \downarrow\\
&= ((\forall x.\,\mathsf{ling}\,x \Rightarrow \mathsf{left}\,x)^{\eta})^{\uparrow} && \textbf{ev.ling}\\
&= \frac{(\forall x.\,\mathsf{ling}\,x \Rightarrow \mathsf{left}\,x)^{\eta}_{p} \multimap [\,]}{p} && \uparrow\\
&= \frac{[\,]}{\forall x.\,\mathsf{ling}\,x \Rightarrow \mathsf{left}\,x} && \text{LeftID}
\end{aligned}$$

Cases with multiple sources of nondeterminism work analogously to the cases examined in Chapter 4. The polymorphism of $\multimap$ allows us to compose $\mathsf{MM}t$ programs with enough structure that multiple sources of indefiniteness can be differentiated post-evaluation. See Fact 5.9. The only substantive difference between this derivation and the State.Set derivation from the selective derivations in Chapter 4 is the underlying monad. Here, as there, there is no obligatory agglomeration of nondeterminism if we do not want it.

Fact 5.9 (Layered $\mathsf{MM}t$ derivation of *a semanticist met a phonologist*).

$$\left(\begin{array}{c}\dfrac{\mathsf{MM}t}{\dfrac{\mathsf{M}t}{e}}\\[2ex] \text{a semanticist}\\[2ex] \dfrac{[\,]}{\dfrac{\textbf{a.sem}_x \multimap [\,]}{x}}\end{array}\left(\begin{array}{cc}\dfrac{\phantom{\mathsf{MM}t}}{\dfrac{\mathsf{M}t}{e \to e \to t}} & \dfrac{\mathsf{MM}t}{\dfrac{\mathsf{M}t}{e}}\\[2ex] \text{met} & \text{a phonologist}\\[2ex] \dfrac{[\,]}{\dfrac{[\,]}{\mathsf{met}}} & \dfrac{\textbf{a.phon}_y \multimap [\,]}{\dfrac{[\,]}{y}}\end{array}\right)\right) = \begin{array}{c}\dfrac{\mathsf{MM}t}{\dfrac{\mathsf{M}t}{t}}\\[2ex] \text{a semanticist met a phonologist}\\[2ex] \dfrac{\textbf{a.phon}_y \multimap [\,]}{\dfrac{\textbf{a.sem}_x \multimap [\,]}{\mathsf{met}\,y\,x}}\end{array}$$

$$\rightsquigarrow \frac{\textbf{a.phon}_y \multimap [\,]}{\textbf{a.sem}_x \multimap (\mathsf{met}\,y\,x)^{\eta}} \quad \downarrow$$

$$\rightsquigarrow \underbrace{\textbf{a.phon}_y \multimap (\textbf{a.sem}_x \multimap (\mathsf{met}\,y\,x)^{\eta})^{\eta}}_{\mathsf{MM}t} \quad \downarrow$$

The semantics will also accommodate a nondeterministic entry for disjunction which is identical in form to the entry given for the State.Set monad, see Definition 5.20 (recall that $m \sqcup n := \lambda s.\, m\, s \cup n\, s$). This guarantees that disjunction takes exceptional scope and gives a solid compositional basis for nondeterministic analyses of disjunction (e.g. Zimmermann 2000, Alonso-Ovalle 2006, Aloni 2007). As in the State.Set monad, this semantics is polymorphic, and so we again automatically predict that disjunctions will be immune to the BRC.

Definition 5.20 (Disjunction in alternative semantics).

$$\frac{\mathsf{M}\beta}{\alpha} \to \frac{\mathsf{M}\beta}{\alpha} \to \frac{\mathsf{M}\beta}{\alpha}$$

or

$$\lambda mnk.\, m\, k \sqcup n\, k$$

In sum, so as far as the nondeterministic scope of indefinites and disjunctions, an underlying Reader.Set monad matches the coverage of an underlying State.Set monad beat for beat, so long as side effects are managed via scope-taking. Just as we inherit the State.Set monad's good predictions for indefinites (exceptional scope-taking, selective scope), so we inherit its good predictions for disjunction (exceptional scope, selective scope, and the apparent inapplicability of the BRC).

5.5.4 Binding

Anaphora presents a major challenge for AltS-style accounts of indefinites. There is no analog of the Derived VP rule for the Reader.Set monad, no suitable semantics for predicate abstraction in a Kratzer & Shimoyama 2002-style alternative semantics, and flipping the layering of the alternative and state-sensitive layers (à la Novel & Romero 2010) means triggering BRC violations.

In spite of all this, managing Reader.Set side effects via scope-taking and continuations allows us to give a robust account of intra-sentential binding. I define a Bind rule for the continuized Reader.Set semantics in Definition 5.21; the towers version is given in Fact 5.10. The Bind rule happens to have *the same form* as the binding rule for an underlying State.Set monad (Definition 3.16). I give some examples of how the Bind rule applies to *a linguist* and *every linguist* in Fact 5.11.

Definition 5.21 (Reader.Set with Bind-shift).

$$m^{\blacktriangleright} := \lambda k.\, m\, (\lambda as.\, k\, a\, \widehat{sa})$$

Fact 5.10 (Bind-shift in the tower notation).

$$\left(\begin{array}{c} \dfrac{\mathsf{M}\beta}{\alpha} \\[2ex] exp \\[2ex] \dfrac{f\,[\,]}{a} \end{array}\right)^{\blacktriangleright} = \begin{array}{c} \dfrac{\mathsf{M}\beta}{\alpha} \\[2ex] exp \\[2ex] \dfrac{f\,[\lambda s.\,[\,]\,\widehat{sa}]}{a} \end{array}$$

Fact 5.11 (Bind-shifting *a linguist* and *every linguist*).

$$\left(\frac{\textbf{a.ling}_x \multimap [\,]}{x}\right)^{\blacktriangleright} = \frac{\textbf{a.ling}_x \multimap \lambda s.\,[\,]\,\widehat{sx}}{x} \qquad \left(\frac{\textbf{ev.ling}\,(\lambda x.\,[\,])}{x}\right)^{\blacktriangleright} = \frac{\textbf{ev.ling}\,(\lambda xs.\,[\,]\,\widehat{sx})}{x}$$

Before going through a test case, I'll introduce a simplification rule to make reasoning about derivations a bit more straightforward. This rule is the Reader.Set correlate of Binding simplification in the State.Set monad. When a pronoun finds itself in the immediate scope of a stack with a newly introduced dref, the pronoun evaluates to that newly introduced dref:

Fact 5.12 (A Reader.Set correlate of binding simplification).

$$\lambda s.\, (\textsf{pro}_v \multimap \pi)\, \widehat{sa} = \lambda s.\, \pi[a/v]\, \widehat{sa}$$

Proof.

$$\begin{aligned}
\lambda s.\, (\textsf{pro}_v \multimap \pi)\, \widehat{sa} &= \lambda s.\, (\lambda s.\, \pi[s_\top/v]\, s)\, \widehat{sa} && \multimap\\
&= \lambda s.\, \pi[\widehat{sa}_\top/v]\, \widehat{sa} && \beta\\
&= \lambda s.\, \pi[a/v]\, \widehat{sa} && \top
\end{aligned}$$

□

Now that the bind rule is in place, I'll show how it allows us to give an AltS-style account of *intra*-sentential binding. We Lift and Bind-shift *a linguist* and Lift *his*. Everything else is composed up as usual: the value-denoting *rubbed* and *head* Lift into scopal expressions via η and $\uparrow$, and we compose everything up via scopal application. A few routine simplifications—most relevantly, an application of the binding simplification rule from Fact 5.12—give us the expected interpretation: **a.ling** sequenced with a trivial program about an individual who rubbed his own head. (A parallel derivation goes through for *every linguist* and as expected yields $(\forall x.\, \textsf{ling}\, x \Rightarrow \textsf{rubbed}\, (\textsf{head}\, x)\, x)^\eta$ on evaluation.)

$$\left(\begin{array}{c}\dfrac{\mathsf{M}t}{e}\\[1ex] \text{a linguist}\\[1ex] \dfrac{\textsf{a.ling}_x \multimap \lambda s.\, [\,]\, \widehat{sx}}{x}\end{array}\left(\begin{array}{cc}\dfrac{\mathsf{M}t}{e \to e \to t} & \dfrac{\mathsf{M}t}{e}\\[1ex] \text{rubbed} & \text{his head}\\[1ex] \dfrac{[\,]}{\textsf{rubbed}} & \dfrac{\textsf{pro}_y \multimap [\,]}{\textsf{head}\, y}\end{array}\right)\right) = \begin{array}{c}\dfrac{\mathsf{M}t}{t}\\[1ex] \text{a linguist rubbed his head}\\[1ex] \dfrac{\textsf{a.ling}_x \multimap \lambda s.\, (\textsf{pro}_y \multimap [\,])\, \widehat{sx}}{\textsf{rubbed}\, (\textsf{head}\, y)\, x}\end{array}$$

$$\begin{array}{rrl}
\rightsquigarrow & \textsf{a.ling}_x \multimap \lambda s.\, (\textsf{pro}_y \multimap (\textsf{rubbed}\, (\textsf{head}\, y)\, x)^\eta)\, \widehat{sx} & \downarrow\\
= & \textsf{a.ling}_x \multimap \lambda s.\, (\textsf{rubbed}\, (\textsf{head}\, x)\, x)^\eta\, \widehat{sx} & \text{Fact 5.12}\\
= & \textsf{a.ling}_x \multimap \lambda s.\, \{\textsf{rubbed}\, (\textsf{head}\, x)\, x\} & \eta,\ \beta\\
= & \textsf{a.ling}_x \multimap (\textsf{rubbed}\, (\textsf{head}\, x)\, x)^\eta & \eta
\end{array}$$

I offer one final derivation. Shan 2004 claims that AltS is especially ill-suited to examples where a quantifier intervenes between a binder and a bind-ee. His example is the question *which painter$_i$ told nobody about which of his$_i$ paintings?*. Here I analyze an analogous sentence with indefinites, namely *a man$_i$ told nobody about a book he$_i$ wrote*, deriving the reading on which *a book he wrote* out-scopes *nobody* (so as to as closely parallel Shan's example as possible). The derivation is notable only for how unremarkable it is. Composition is straightforward, the pronoun gets bound, and Lowering yields the expected monadic meaning:

$$\left(\begin{array}{c} \begin{array}{c} \mathsf{M}t \\ \hline \mathsf{M}t \\ \hline e \end{array} \\ \\ \text{a man} \\ \\ \begin{array}{c} \textbf{a.man}_x \multimap \lambda s.\, [\,]\, \widehat{sx} \\ \hline [\,] \\ \hline x \end{array} \end{array} \left(\begin{array}{c} \begin{array}{c} \mathsf{M}t \\ \hline \mathsf{M}t \\ \hline e \to e \to t \end{array} \\ \\ \text{told nobody about} \\ \\ \begin{array}{c} [\,] \\ \hline \textbf{nb}\, (\lambda y.\, [\,]) \\ \hline \lambda z.\, \mathsf{told}\, y\, (\mathsf{about}\, z) \end{array} \end{array} \begin{array}{c} \begin{array}{c} \mathsf{M}t \\ \hline \mathsf{M}t \\ \hline e \end{array} \\ \\ \text{a book he wrote} \\ \\ \begin{array}{c} \textbf{pro}_u \multimap (\textbf{a.bk.by}\, u)_z \multimap [\,] \\ \hline [\,] \\ \hline z \end{array} \end{array}\right)\right)$$

$$\begin{array}{clr} \rightsquigarrow & \textbf{a.man}_x \multimap \lambda s.\, (\textbf{pro}_u \multimap (\textbf{a.bk.by}\, u)_z \multimap \textbf{nb}\, (\lambda y.\, (\mathsf{told}\, y\, (\mathsf{about}\, z)\, x)^{\eta}))\, \widehat{sx} & \mathbb{S}, \downdownarrows \\ = & \textbf{a.man}_x \multimap \lambda s.\, ((\textbf{a.bk.by}\, x)_z \multimap \textbf{nb}\, (\lambda y.\, (\mathsf{told}\, y\, (\mathsf{about}\, z)\, x)^{\eta}))\, \widehat{sx} & \text{Fact 5.12} \\ = & \textbf{a.man}_x \multimap (\textbf{a.bk.by}\, x)_z \multimap \textbf{nb}\, (\lambda y.\, (\mathsf{told}\, y\, (\mathsf{about}\, z)\, x)^{\eta}) & \eta,\ \beta \\ = & \textbf{a.man}_x \multimap (\textbf{a.bk.by}\, x)_z \multimap (\neg\exists y.\, \mathsf{told}\, y\, (\mathsf{about}\, z)\, x)^{\eta} & \textbf{nb} \end{array}$$

Of course, we should assure ourselves that none of this comes at the expense of BRC violations. Clearly, however, because nondeterminism acquires scope via bona fide scope mechanisms, we respect the BRC for the same reasons detailed for the State.Set monad in Chapter 4.

All told, using scope to propagate Reader.Set side effects gives a robust account of *intra-sentential* anaphora. But there's a price: using the Reader.Set monad in lieu of the State.Set monad means giving up our account of cross-sentential (and donkey) anaphora. Since the Reader.Set's η operation ($\lambda as.\, \{a\}$) is designed to take a stack and discard it (whereas the State.Set's η holds onto a stack in a pair), binding potential inevitably disappears with evaluation. Notice that neither the derivation for *a linguist*$_i$ *rubbed his*$_i$ *head* nor the derivation for *a man*$_i$ *told nobody about a book he*$_i$ *wrote* retain any trace of the modified stack post-evaluation. Thus, while re-Lifting will allow the indefinite to take "quantificational" (i.e. nondeterministic) scope post-evaluation, the updated stack used for securing binding has disappeared, never to return:

$$(\textbf{a.ling}_x \multimap (\mathsf{rubbed}\, (\mathsf{head}\, x)\, x)^{\eta})^{\uparrow} = \frac{\textbf{a.ling}_x \multimap [\,]}{\mathsf{rubbed}\, (\mathsf{head}\, x)\, x}$$

Thus, though the Reader.Set monad allows indefinites to project their existential force (i.e. nondeterminism) out of scope islands, it does not allow them to bind out of scope islands. Reader.Set programs, like Reader programs, just aren't build to hold onto anaphoric potential. The incoming stack is used only to calculate a value and is promptly discarded as soon as that happens; as soon as we evaluate, our anaphoric goose is cooked. (Incidentally, we have also lost a bit of explanatory force. Whereas before, we could classify increasing quantifiers into exceptional scope-takers or not by the ways in which they bind pronouns outside of their scopes—or vice versa—this option disappears here. Whether something behaves like an indefinite will be an item-by-item stipulation, or will have to be determined via other means—e.g. morphology, syntax.)

The Reader.Set and State.Set monads differ only in the outer monad layered around the nondeterministic Set monad: the Reader.Set monad is derived by applying the ReaderT monad transformer to the Set monad, while the State.Set monad by applying the StateT monad transformer to the Set monad. This offers in my view an exceptionally clear reckoning with two things: first, of the precise way in which DyS-style theorizing about indefinites and AltS-style theorizing about indefinites differ—namely, *State*T versus *Reader*T—and second, of the precise thing which DyS-style theorizing and AltS-style theorizing about indefiniteness *share*—namely, both rely on the Set monad to give a nondeterministic treatment of indefiniteness. The account of exceptional existential scope-taking is rooted in the Set monad (which Reader.Set and State.Set share), while the account of exceptional binding configurations is rooted in the State monad (which Reader.Set naturally lacks).

5.6 Conclusion

This chapter provided additional motivation for the account of exceptional scope-taking defended in Chapter 4. The first two case studies focused on anaphoric side effects. It was shown that the theory of exceptional scope—in terms of side effects surviving and taking scope after evaluation—makes a number of non-obvious predictions which seem to be a good fit with the data, and which aren't characteristic of other treatments of exceptional scope (which are generally concerned with indefinites and indefiniteness to the exclusion of anaphoric phenomena). Specifically, I detailed how the theory of exceptional scope explains the existence of both (cross-categorial) surprising sloppy readings and maximal discourse reference to deeply embedded restrictors and dynamic GQs. In general, the theory predicts exceptional scope to be far more common and anaphoric processes to be far less constrained than extant theories, a prediction I argue is met with a fair amount of empirical support.

Moving on from anaphora, I offered an account of focus qua side effect which explains both the island-insensitivity and potential selectivity of AWF. This required nothing beyond refactoring Rooth's 1985 theory of focus interpretation in terms of side effects using the Focus monad. In addition to offering an explanation for some recalcitrant data, the account illuminated a deep and thoroughgoing connection between the ways in which focus alternatives and indefinites take scope out of islands.

I concluded with a novel compositional treatment of alternative semantics in terms of the Reader.Set monad. I argued that the theory held up better than Kratzer & Shimoyama's 2002 formulation, while staying close to their account's spirit and motivations. The account underlined how the explanation of exceptional nondeterministic scope-taking is general and does not turn on the precise underlying monad presumed in Chapter 4 (namely State.Set). In addition, the monadic perspective illuminated a fundamental and (so far as I know) unremarked connection between dynamic and alternative semantics for indefinites (both are nondeterministic, reflected in their reliance on the Set monad), while providing a clear accounting of the way in which they differ (their divergent perspectives on anaphora correspond to a choice between the Reader and State monads).

Appendix A

Fragment

A.1 Types

Definition A.1 (Types). Given a monadic type constructor M:

$$T ::= e \mid t \mid T/T \mid T\backslash T \mid \mathsf{M}T \mid \langle T \rangle \qquad \mathsf{K}_R T ::= \mathsf{M}R/(\mathsf{M}R/T)$$

Definition A.2 (Slash elimination rules).

$$\frac{\Gamma \vdash F : T/S \quad \Delta \vdash E : S}{\Gamma \cdot \Delta \vdash F E : T}\, / \qquad \frac{\Delta \vdash E : S \quad \Gamma \vdash F : S\backslash T}{\Delta \cdot \Gamma \vdash F E : T}\, \backslash$$

A.2 Combinators

Definition A.3 (Monadic combinators). Given some ε such that $\varepsilon \cdot \Gamma = \Gamma \cdot \varepsilon = \Gamma$:

$$\frac{}{\varepsilon \vdash \eta : \mathsf{M}T/T}\,\text{Ax} \qquad \frac{}{\varepsilon \vdash (\multimap) : \mathsf{K}_R T/\mathsf{M}T}\,\text{Ax}$$

Definition A.4 (The monad laws).

$$\frac{\Gamma \vdash E[a_v^\eta \multimap \pi] : T}{\Gamma \vdash E[\pi[a/v]] : T}\,\text{LeftID} \qquad \frac{\Gamma \vdash E[m_v \multimap v^\eta] : T}{\Gamma \vdash E[m] : T}\,\text{RightID} \qquad \frac{\Gamma \vdash E[(m_v \multimap \pi)_u \multimap \theta] : T}{\Gamma \vdash E[m_v \multimap \pi_u \multimap \theta] : T}\,\text{Assoc}$$

Definition A.5 (Scopal combinators).

$$\frac{}{\varepsilon \vdash \mathbf{S}_/ : (\mathsf{K}_R T/\mathsf{K}_R S)/\mathsf{K}_R(T/S)}\,\text{Ax} \qquad \mathbf{S}_/ F\, E := \lambda k.\, F\, (\lambda f.\, E\, (\lambda x.\, k\, (f\, x)))$$

$$\frac{}{\varepsilon \vdash \mathbf{S}_\backslash : (\mathsf{K}_R T/\mathsf{K}_R(S\backslash T))/\mathsf{K}_R S}\,\text{Ax} \qquad \mathbf{S}_\backslash E\, F := \lambda k.\, E\, (\lambda x.\, F\, (\lambda f.\, k\, (f\, x)))$$

Definition A.6 (Unary/binary notation for combinators).

$$\frac{\Gamma \vdash E : T}{\Gamma \vdash E^\eta : \mathsf{M}T}\,\eta \qquad \frac{\Gamma \vdash E : \mathsf{M}T}{\Gamma \vdash (\multimap)\, E : \mathsf{K}_R T}\,\multimap$$

$$\frac{\Gamma \vdash F : \mathsf{K}_R(T/S) \quad \Delta \vdash E : \mathsf{K}_R S}{\Gamma \cdot \Delta \vdash \mathbf{S}_/ F\, E : \mathsf{K}_R T}\,\mathbf{S}_/ \qquad \frac{\Delta \vdash E : \mathsf{K}_R S \quad \Gamma \vdash F : \mathsf{K}_R(S\backslash T)}{\Delta \cdot \Gamma \vdash \mathbf{S}_\backslash E\, F : \mathsf{K}_R T}\,\mathbf{S}_\backslash$$

A.3 Deriving ancillary operations

Fact A.1 (External lift).

$$\Gamma \vdash a : T \Rightarrow \Gamma \vdash \lambda k.\, k\, a : \mathsf{K}_R T$$

Proof.

$$\dfrac{\dfrac{\dfrac{}{\varepsilon \vdash (\multimap) : \mathsf{K}_R T/\mathsf{M}T}\,\text{Ax} \quad \dfrac{\dfrac{}{\varepsilon \vdash \eta : \mathsf{M}T/T}\,\text{Ax} \quad \Gamma \vdash a : T}{\Gamma \vdash a^{\eta} : \mathsf{M}T}\,/}{\Gamma \vdash \lambda k.\, a^{\eta} \multimap k : \mathsf{K}_R T}\,/}{\Gamma \vdash \lambda k.\, k\, a : \mathsf{K}_R T}\,\text{LeftID}$$

□

Fact A.2 (Internal Lift).

$$\Gamma \vdash m : \mathsf{K}_R T \Rightarrow \Gamma \vdash \lambda c.\, m\, (\lambda a.\, c\, (\lambda k.\, k\, a)) : \mathsf{K}_R(\mathsf{K}_S T)$$

Proof.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\varepsilon \vdash (\multimap) : \mathsf{K}_S T/\mathsf{M}T}\,\text{Ax}}{\varepsilon \vdash \lambda c.\, c\, (\multimap) : \mathsf{K}_R(\mathsf{K}_S T/\mathsf{M}T)}\,\text{FA.1} \quad \dfrac{\dfrac{\dfrac{}{\varepsilon \vdash \eta : \mathsf{M}T/T}\,\text{Ax}}{\varepsilon \vdash \lambda c.\, c\, \eta : \mathsf{K}_R(\mathsf{M}T/T)}\,\text{FA.1} \quad \Gamma \vdash m : \mathsf{K}_R T}{\Gamma \vdash \lambda c.\, m\, (\lambda a.\, c\, a^{\eta}) : \mathsf{K}_R \mathsf{M}T}\,\mathbf{S}_{/}}{\Gamma \vdash \lambda c.\, m\, (\lambda a.\, c\, (\lambda k.\, a^{\eta} \multimap k)) : \mathsf{K}_R(\mathsf{K}_S T)}\,\mathbf{S}_{/}}{\Gamma \vdash \lambda c.\, m\, (\lambda a.\, c\, (\lambda k.\, k\, a)) : \mathsf{K}_R(\mathsf{K}_S T)}\,\text{LeftID}$$

□

Fact A.3 (3-level combination).

$$\Gamma \vdash M : \mathsf{K}_U(\mathsf{K}_R S),\ \Delta \vdash N : \mathsf{K}_U(\mathsf{K}_R(S\backslash T)) \Rightarrow$$
$$\Gamma \cdot \Delta \vdash \lambda c.\, M\, (\lambda m.\, N\, (\lambda n.\, c\, (\mathbf{S}_{\backslash} m\, n))) : \mathsf{K}_U(\mathsf{K}_R T)$$

Proof.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\varepsilon \vdash \mathbf{S}_{\backslash} : (\mathsf{K}_R T/\mathsf{K}_R(S\backslash T))/\mathsf{K}_R S}\,\text{Ax}}{\varepsilon \vdash \lambda k.\, k\, \mathbf{S}_{\backslash} : \mathsf{K}_U((\mathsf{K}_R T/\mathsf{K}_R(S\backslash T))/\mathsf{K}_R S)}\,\text{FA.1} \quad \Gamma \vdash M : \mathsf{K}_U(\mathsf{K}_R S)}{\Gamma \vdash \lambda c.\, M\, (\lambda m.\, c\, (\mathbf{S}_{\backslash} m)) : \mathsf{K}_U(\mathsf{K}_R T/\mathsf{K}_R(S\backslash T))}\,\mathbf{S}_{/} \quad \Delta \vdash N : \mathsf{K}_U(\mathsf{K}_R(S\backslash T))}{\Gamma \cdot \Delta \vdash \lambda c.\, M\, (\lambda m.\, N\, (\lambda n.\, c\, (\mathbf{S}_{\backslash} m\, n))) : \mathsf{K}_U(\mathsf{K}_R T)}\,\mathbf{S}_{/}$$

□

$$\Gamma \vdash M : \mathsf{K}_U(\mathsf{K}_R(T/S)),\ \Delta \vdash N : \mathsf{K}_U(\mathsf{K}_R S) \Rightarrow$$
$$\Gamma \cdot \Delta \vdash \lambda c.\, M\, (\lambda m.\, N\, (\lambda n.\, c\, (\mathbf{S}_{/} m\, n))) : \mathsf{K}_U(\mathsf{K}_R T)$$

Proof.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\varepsilon \vdash \mathbf{S}_{/} : (\mathsf{K}_R T/\mathsf{K}_R S)/\mathsf{K}_R(T/S)}\,\text{Ax}}{\varepsilon \vdash \lambda k.\, k\, \mathbf{S}_{/} : \mathsf{K}_U((\mathsf{K}_R T/\mathsf{K}_R S)/\mathsf{K}_R(T/S))}\,\text{FA.1} \quad \Gamma \vdash M : \mathsf{K}_U(\mathsf{K}_R(T/S))}{\Gamma \vdash \lambda c.\, M\, (\lambda m.\, c\, (\mathbf{S}_{/} m)) : \mathsf{K}_U(\mathsf{K}_R T/\mathsf{K}_R S)}\,\mathbf{S}_{/} \quad \Delta \vdash N : \mathsf{K}_U(\mathsf{K}_R S)}{\Gamma \cdot \Delta \vdash \lambda c.\, M\, (\lambda m.\, N\, (\lambda n.\, c\, (\mathbf{S}_{/} m\, n))) : \mathsf{K}_U(\mathsf{K}_R T)}\,\mathbf{S}_{/}$$

□

Fact A.4 (2-level Lower).

$$\Gamma \vdash M : \mathsf{K}_R R \Rightarrow \Gamma \vdash M\,\eta : \mathsf{M}R$$

Proof.

$$\dfrac{\dfrac{\Gamma \vdash M : \mathsf{K}_R R}{\Gamma \vdash M : \mathsf{M}R/(\mathsf{M}R/R)}\,\text{DA.1} \quad \dfrac{}{\varepsilon \vdash \eta : \mathsf{M}R/R}\,\text{Ax}}{\Gamma \vdash M\,\eta : \mathsf{M}R}\,/$$

□

Fact A.5 (Internal Lower).

$$\Gamma \vdash M : \mathsf{K}_S(\mathsf{K}_R R) \Rightarrow \Gamma \vdash \lambda k.\, M\,(\lambda m.\, k\,(m\,\eta)) : \mathsf{K}_S \mathsf{M}R$$

Proof.

$$\dfrac{\dfrac{\Gamma \vdash M : \mathsf{K}_S(\mathsf{K}_R R)}{\Gamma \vdash M : \mathsf{K}_S(\mathsf{M}R/(\mathsf{M}R/R))}\,\text{DA.1} \quad \dfrac{\dfrac{}{\varepsilon \vdash \eta : \mathsf{M}R/R}\,\text{Ax}}{\varepsilon \vdash \lambda k.\, k\,\eta : \mathsf{K}_S(\mathsf{M}R/R)}\,\text{FA.1}}{\Gamma \vdash \lambda k.\, M\,(\lambda m.\, k\,(m\,\eta)) : \mathsf{K}_S \mathsf{M}R}\,\mathbf{S}_/$$

□

Fact A.6 (3-level Lower).

$$\Gamma \vdash M : \mathsf{K}_R(\mathsf{K}_R R) \Rightarrow \Gamma \vdash M\,(\lambda m.\, m\,\eta) : \mathsf{M}R$$

Proof.

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash M : \mathsf{K}_R(\mathsf{K}_R R)}{\Gamma \vdash M : \mathsf{K}_R(\mathsf{M}R/(\mathsf{M}R/R))}\,\text{DA.1}}{\Gamma \vdash M : \mathsf{M}R/(\mathsf{M}R/(\mathsf{M}R/(\mathsf{M}R/R)))}\,\text{DA.1} \quad \dfrac{\dfrac{\dfrac{}{\varepsilon \vdash \eta : \mathsf{M}R/R}\,\text{Ax}}{\varepsilon \vdash \lambda m.\, m\,\eta : \mathsf{K}_R(\mathsf{M}R/R)}\,\text{FA.1}}{\varepsilon \vdash \lambda m.\, m\,\eta : \mathsf{M}R/(\mathsf{M}R/(\mathsf{M}R/R))}\,\text{DA.1}}{\Gamma \vdash M\,(\lambda m.\, m\,\eta) : \mathsf{M}R}\,/$$

□

A.4 Islands

Definition A.7 (Type occurrences). The type T is a positive occurrence in T. If T/S or $S\backslash T$ is a positive (resp. negative) occurrence, S is a negative (positive) occurrence, and T is a positive (negative) occurrence. If $\mathsf{M}T$ is a positive (negative) occurrence, T is a positive (negative) occurrence.

Definition A.8 (Unevaluated types). Given an M determining a set of types T, the set of unevaluated types $N \subseteq T$, is the smallest set such that: (i) for any $S \in T$ and any $R \in T$, $\mathsf{K}_R S \in N$; (ii) for any $S \in T$, if there is a positive occurrence of an unevaluated type in S, $S \in N$.

Note. Requiring *positive* occurrences rules out cases like $S/\mathsf{K}_R T$ and rules in cases like $\mathsf{K}_R T/S$ (where $S \notin N$). Only positive occurrences of a type indicate the "presence" of a value of that type.

Definition A.9 (Evaluated types). Given a set of types T and a set of unevaluated types $N \subseteq T$:

$$E ::= T - N$$

Definition A.10 (Islands and enforcing evaluation).

$$\dfrac{\Gamma \vdash m : T \in E}{\Gamma \vdash m : \langle T\rangle}\,\text{Eval} \qquad \dfrac{}{\mathrm{fin} \vdash \blacksquare : T/\langle T\rangle}\,\text{Lex} \qquad \blacksquare E \coloneqq E$$

Definition A.11 (Scope islands). A scope island is any constituent that is the sister of fin.

Fact A.7. Side effects scope out of scope islands.

Proof. Supposing that $\mathsf{M}R \in E$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\varepsilon \vdash (\multimap) : \mathsf{K}_S R/\mathsf{M}R}\,\text{Ax} \quad \dfrac{\dfrac{}{\mathrm{fin} \vdash \blacksquare : \mathsf{M}R/\langle \mathsf{M}R\rangle}\,\text{Lex} \quad \dfrac{\dfrac{\Gamma \vdash \lambda k.\, m_\nu \multimap k\,(f\,\nu) : \mathsf{K}_R R}{\Gamma \vdash m_\nu \multimap (f\,\nu)^\eta : \mathsf{M}R}\,\text{FA.4}}{\Gamma \vdash m_\nu \multimap (f\,\nu)^\eta : \langle \mathsf{M}R\rangle}\,\text{Eval}}{\mathrm{fin}\cdot\Gamma \vdash m_\nu \multimap (f\,\nu)^\eta : \mathsf{M}R}\,/}{\mathrm{fin}\cdot\Gamma \vdash \lambda k.\, (m_\nu \multimap (f\,\nu)^\eta)_u \multimap k\,u : \mathsf{K}_S R}\,/}{\mathrm{fin}\cdot\Gamma \vdash \lambda k.\, m_\nu \multimap (f\,\nu)^\eta_u \multimap k\,u : \mathsf{K}_S R}\,\text{Assoc}}{\mathrm{fin}\cdot\Gamma \vdash \lambda k.\, m_\nu \multimap k\,(f\,\nu) : \mathsf{K}_S R}\,\text{LeftID}$$

□

A.5 Sequences

Definition A.12 (Sequences and operations on sequences).

$$s \Coloneqq \text{the set of all sequences of values} \qquad \text{if } s = \alpha\beta\cdots\omega,\ \widehat{sE} \coloneqq \alpha\beta\cdots\omega E$$

$$s_\top \coloneqq \text{the last element of } s$$

A.6 The State.Set monad

Definition A.13 (The State.Set monad).

$$\begin{aligned} \mathsf{M}\alpha &\Coloneqq s \to (\alpha \times s) \to t \\ a^\eta &\coloneqq \lambda s.\, \{\langle a,\, s\rangle\} \\ m_\nu \multimap \pi &\coloneqq \lambda s. \bigcup_{\langle a, s'\rangle \in m\,s} \pi[a/\nu]\, s' \end{aligned}$$

Definition A.14 (State.Set lexicon sampling).

Item	**Meaning**	**Type**
John	j	e
saw	saw	$(e\backslash t)/e$
and	$(\wedge)$	$(t\backslash t)/t$
a linguist	$\textbf{a.ling} \coloneqq \lambda s.\, \{\langle x,\, s\rangle \mid \mathsf{ling}\, x\}$	$\mathsf{M}e$
she, GAP	$\textbf{pro} \coloneqq \lambda s.\, \{\langle s_\top,\, s\rangle\}$	$\mathsf{M}e$
not	$\textbf{not} \coloneqq \lambda m s.\, \{\langle \neg\exists s'.\, \langle \mathsf{True},\, s'\rangle \in m\, s,\ s\rangle\}$	$\mathsf{M}t/\mathsf{M}t$
if	$\textbf{if} \coloneqq \lambda m n.\, \textbf{not}\, (m_p \multimap (\textbf{not}\, n)_q \multimap (p \wedge q)^\eta)$	$(\mathsf{M}t/\mathsf{M}t)/\mathsf{M}t$
every linguist	$\textbf{ev.ling} \coloneqq \lambda k.\, \textbf{not}\, ((\textbf{a.ling})_x \multimap \textbf{not}\, (k\, x))$	$\mathsf{K}_t\, e$
or	$\textbf{or} \coloneqq \lambda m n k s.\, m\, k\, s \cup n\, k\, s$	$(\mathsf{K}_R T\backslash \mathsf{K}_R T)/\mathsf{K}_R T$

Definition A.15 (State.Set dref introduction).

$$m^{\triangleright} := m_{\nu} \multimap \lambda as.\, a^{\eta}\widehat{sa} \qquad \frac{\Gamma \vdash \lambda k.\, E[m_{\nu} \multimap k\, \nu] : \mathsf{K}_R T}{\Gamma \vdash \lambda k.\, E[m^{\triangleright}_{\nu} \multimap k\, \nu] : \mathsf{K}_R T}\,\blacktriangleright \qquad \frac{\Gamma \vdash E[m^{\triangleright}_{\nu} \multimap \mathbf{pro}_u \multimap \pi] : T}{\Gamma \vdash E[m^{\triangleright}_{\nu} \multimap \pi[\nu/u]] : T}\,\text{Binding}$$

Fact A.8. Surface scope for *a man met every linguist* (semantics only).

Proof.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{a.man} : \mathsf{M}e}\,\text{Lex}}{\lambda k.\, \mathbf{a.man}_x \multimap k\, x : \mathsf{K}_t e}\,\multimap \qquad \dfrac{\dfrac{\dfrac{}{\mathsf{met} : (e\backslash t)/e}\,\text{Lex}}{\lambda k.\, k\, \mathsf{met} : \mathsf{K}_t((e\backslash t)/e)}\,\text{FA.1} \quad \dfrac{}{\mathbf{ev.ling} : \mathsf{K}_t e}\,\text{Lex}}{\lambda k.\, \mathbf{ev.ling}\,(\lambda y.\, k\,(\mathsf{met}\, y)) : \mathsf{K}_t(e\backslash t)}\,\mathbf{S}_{/}}{\dfrac{\lambda k.\, \mathbf{a.man}_x \multimap \mathbf{ev.ling}\,(\lambda y.\, k\,(\mathsf{met}\, y\, x)) : \mathsf{K}_t t}{\mathbf{a.man}_x \multimap \mathbf{ev.ling}\,(\lambda y.\, (\mathsf{met}\, y\, x)^{\eta}) : \mathsf{M}t}\,\text{FA.4}}\,\mathbf{S}_{\backslash}}{\mathbf{a.man}_x \multimap (\forall y.\, \mathsf{ling}\, y \Rightarrow \mathsf{met}\, y\, x)^{\eta} : \mathsf{M}t}\,\mathbf{ev.ling}$$

□

Fact A.9. Inverse scope for *a man met every linguist*.

Proof.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{a.man} : \mathsf{M}e}\,\text{Lex}}{\lambda k.\, \mathbf{a.man}_x \multimap k\, x : \mathsf{K}_t e}\,\multimap}{\lambda c.\, c\,(\lambda k.\, \mathbf{a.man}_x \multimap k\, x) : \mathsf{K}_t(\mathsf{K}_t e)}\,\text{FA.1} \quad \dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{met} : (e\backslash t)/e}\,\text{Lex}}{\lambda k.\, k\, \mathsf{met} : \mathsf{K}_t((e\backslash t)/e)}\,\text{FA.1} \quad \dfrac{}{\mathbf{ev.ling} : \mathsf{K}_t e}\,\text{Lex}}{\lambda k.\, \mathbf{ev.ling}\,(\lambda y.\, k\,(\mathsf{met}\, y)) : \mathsf{K}_t(e\backslash t)}\,\mathbf{S}_{/}}{\lambda c.\, \mathbf{ev.ling}\,(\lambda y.\, c\,(\lambda k.\, k\,(\mathsf{met}\, y))) : \mathsf{K}_t(\mathsf{K}_t(e\backslash t))}\,\text{FA.2}}{\lambda c.\, \mathbf{ev.ling}\,(\lambda y.\, c\,(\lambda k.\, \mathbf{a.man}_x \multimap k\,(\mathsf{met}\, y\, x))) : \mathsf{K}_t(\mathsf{K}_t(e\backslash t))}\,\text{FA.3}}{\mathbf{ev.ling}\,(\lambda y.\, \mathbf{a.man}_x \multimap (\mathsf{met}\, y\, x)^{\eta}) : \mathsf{M}t}\,\text{FA.6}}{(\forall y.\, \mathsf{ling}\, y \Rightarrow \exists x.\, \mathsf{man}\, x \wedge \mathsf{met}\, y\, x)^{\eta} : \mathsf{M}t}\,\mathbf{ev.ling}, \mathbf{a.man}$$

□

Fact A.10. Indefiniteness scopes out of ⟨*A man met every linguist*⟩ on the surface-scope derivation. Nothing scopes out on the inverse-scope derivation.

Proof.

$$\dfrac{\dfrac{\dfrac{}{\blacksquare : \mathsf{M}t/\langle \mathsf{M}t\rangle}\,\text{Lex} \quad \dfrac{\dfrac{\vdots}{\mathbf{a.man}_x \multimap (\forall y.\, \mathsf{ling}\, y \Rightarrow \mathsf{met}\, y\, x)^{\eta} : \mathsf{M}t}\,\text{FA.8}}{\mathbf{a.man}_x \multimap (\forall y.\, \mathsf{ling}\, y \Rightarrow \mathsf{met}\, y\, x)^{\eta} : \langle \mathsf{M}t\rangle}\,\text{Eval}}{\mathbf{a.man}_x \multimap (\forall y.\, \mathsf{ling}\, y \Rightarrow \mathsf{met}\, y\, x)^{\eta} : \mathsf{M}t}\,/}{\lambda k.\, \mathbf{a.man}_x \multimap k\,(\forall y.\, \mathsf{ling}\, y \Rightarrow \mathsf{met}\, y\, x) : \mathsf{K}_t t}\,\multimap$$

$$\dfrac{\dfrac{\dfrac{}{\blacksquare : \mathsf{M}t/\langle \mathsf{M}t\rangle}\,\text{Lex} \quad \dfrac{\dfrac{\vdots}{(\forall y.\, \mathsf{ling}\, y \Rightarrow \exists x.\, \mathsf{man}\, x \wedge \mathsf{met}\, y\, x)^{\eta} : \mathsf{M}t}\,\text{FA.9}}{(\forall y.\, \mathsf{ling}\, y \Rightarrow \exists x.\, \mathsf{man}\, x \wedge \mathsf{met}\, y\, x)^{\eta} : \langle \mathsf{M}t\rangle}\,\text{Eval}}{(\forall y.\, \mathsf{ling}\, y \Rightarrow \exists x.\, \mathsf{man}\, x \wedge \mathsf{met}\, y\, x)^{\eta} : \mathsf{M}t}\,/}{\lambda k.\, k\,(\forall y.\, \mathsf{ling}\, y \Rightarrow \exists x.\, \mathsf{man}\, x \wedge \mathsf{met}\, y\, x) : \mathsf{K}_t t}\,\multimap$$

□

Fact A.11. Deriving ⟨*a man$_i$ walked*⟩; ⟨*he$_i$ hummed*⟩.

Proof.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{a.man} : \mathsf{M}e}\,\text{Lex}}{\lambda k.\,\mathbf{a.man}_x \multimap k\,x : \mathsf{K}_t e}\multimap}{\lambda k.\,\mathbf{a.man}^{\triangleright}_x \multimap k\,x : \mathsf{K}_t e}\blacktriangleright \quad \dfrac{\dfrac{}{\mathsf{w} : e\backslash t}\,\text{Lex}}{\lambda k.\,k\,\mathsf{w} : \mathsf{K}_t(e\backslash t)}\,\text{FA.1}}{\lambda k.\,\mathbf{a.man}^{\triangleright}_x \multimap k\,(\mathsf{w}\,x) : \mathsf{K}_t t}\,\mathbf{S}_{\backslash}}{\mathbf{a.man}^{\triangleright}_x \multimap (\mathsf{w}\,x)^{\eta} : \mathsf{M}t}\,\text{FA.4}}{\mathbf{a.man}^{\triangleright}_x \multimap (\mathsf{w}\,x)^{\eta} : \langle\mathsf{M}t\rangle}\,\text{Eval}}{}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\blacksquare : \mathsf{M}t/\langle\mathsf{M}t\rangle}\,\text{Lex} \quad \mathbf{a.man}^{\triangleright}_x \multimap (\mathsf{w}\,x)^{\eta} : \langle\mathsf{M}t\rangle}{\mathbf{a.man}^{\triangleright}_x \multimap (\mathsf{w}\,x)^{\eta} : \mathsf{M}t}\,/}{\lambda k.\,\mathbf{a.man}^{\triangleright}_x \multimap k\,(\mathsf{w}\,x) : \mathsf{K}_t t}\multimap \quad \dfrac{\dfrac{\dfrac{}{(\wedge) : (t\backslash t)/t}\,\text{Lex}}{\lambda k.\,k\,(\wedge) : \mathsf{K}_t((t\backslash t)/t)}\,\text{FA.1} \quad \lambda k.\,\mathbf{pro}_y \multimap k\,(\mathsf{h}\,y) : \mathsf{K}_t t}{\lambda k.\,\mathbf{pro}_y \multimap k\,(\lambda p.\,p \wedge \mathsf{h}\,y) : \mathsf{K}_t(t\backslash t)}\,\mathbf{S}_{/}}{\lambda k.\,\mathbf{a.man}^{\triangleright}_x \multimap \mathbf{pro}_y \multimap k\,(\mathsf{w}\,x \wedge \mathsf{h}\,y) : \mathsf{K}_t t}\,\mathbf{S}_{\backslash}}{\lambda k.\,\mathbf{a.man}^{\triangleright}_x \multimap k\,(\mathsf{w}\,x \wedge \mathsf{h}\,x) : \mathsf{K}_t t}\,\text{Binding}$$

where

$$\dfrac{\dfrac{}{\blacksquare : \mathsf{M}t/\langle\mathsf{M}t\rangle}\,\text{Lex} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{pro} : \mathsf{M}e}\,\text{Lex}}{\lambda k.\,\mathbf{pro}_y \multimap k\,y : \mathsf{K}_t e}\multimap \quad \dfrac{\dfrac{}{\mathsf{h} : e\backslash t}\,\text{Lex}}{\lambda k.\,k\,\mathsf{h} : \mathsf{K}_t(e\backslash t)}\,\text{FA.1}}{\lambda k.\,\mathbf{pro}_y \multimap k\,(\mathsf{h}\,y) : \mathsf{K}_t t}\,\mathbf{S}_{\backslash}}{\mathbf{pro}_y \multimap (\mathsf{h}\,y)^{\eta} : \mathsf{M}t}\,\text{FA.4}}{\mathbf{pro}_y \multimap (\mathsf{h}\,y)^{\eta} : \langle\mathsf{M}t\rangle}\,\text{Eval}}{\dfrac{\mathbf{pro}_y \multimap (\mathsf{h}\,y)^{\eta} : \mathsf{M}t}{\lambda k.\,\mathbf{pro}_y \multimap k\,(\mathsf{h}\,y) : \mathsf{K}_t t}\multimap}\,/$$

□

Fact A.12. Deriving *a* > *if* reading of *if* ⟨*a relative of mine dies*⟩, ⟨*I'll be rich*⟩.

Proof.

$$\dfrac{\dfrac{\dfrac{}{\blacksquare : \mathsf{M}t/\langle\mathsf{M}t\rangle}\,\text{Lex} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{a.rel} : \mathsf{M}e}\,\text{Lex}}{\lambda k.\,\mathbf{a.rel}_x \multimap k\,x : \mathsf{K}_t e}\multimap \quad \dfrac{\dfrac{}{\mathsf{d} : e\backslash t}\,\text{Lex}}{\lambda k.\,k\,\mathsf{d} : \mathsf{K}_t(e\backslash t)}\,\text{FA.1}}{\lambda k.\,\mathbf{a.rel}_x \multimap k\,(\mathsf{d}\,x) : \mathsf{K}_t t}\,\mathbf{S}_{\backslash}}{\mathbf{a.rel}_x \multimap (\mathsf{d}\,x)^{\eta} : \mathsf{M}t}\,\text{FA.4}}{\mathbf{a.rel}_x \multimap (\mathsf{d}\,x)^{\eta} : \langle\mathsf{M}t\rangle}\,\text{Eval}}{\mathbf{a.rel}_x \multimap (\mathsf{d}\,x)^{\eta} : \mathsf{M}t}\,/$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{if} : (\mathsf{M}t/\mathsf{M}t)/\mathsf{M}t}\,\text{Lex}}{\mathbf{if} : \mathsf{K}_t((\mathsf{M}t/\mathsf{M}t)/\mathsf{M}t)}\,\text{FA.1} \quad \dfrac{\dfrac{\dfrac{}{\eta : \mathsf{M}t/t}\,\text{Ax}}{\lambda k.\,k\,\eta : \mathsf{K}_t(\mathsf{M}t/t)}\,\text{FA.1} \quad \dfrac{\mathbf{a.rel}_x \multimap (\mathsf{d}\,x)^{\eta} : \mathsf{M}t}{\lambda k.\,\mathbf{a.rel}_x \multimap k\,(\mathsf{d}\,x) : \mathsf{K}_t t}\multimap}{\lambda k.\,\mathbf{a.rel}_x \multimap k\,(\mathsf{d}\,x)^{\eta} : \mathsf{K}_t\mathsf{M}t}\,\mathbf{S}_{/}}{\lambda k.\,\mathbf{a.rel}_x \multimap k\,(\mathbf{if}\,(\mathsf{d}\,x)^{\eta}) : \mathsf{K}_t(\mathsf{M}t/\mathsf{M}t)}\,\mathbf{S}_{/}}{}$$

$$\dfrac{\dfrac{\dfrac{}{\blacksquare : t/\langle t\rangle}\,\text{Lex} \quad \dfrac{\dfrac{\dfrac{}{\mathsf{i} : e}\,\text{Lex} \quad \dfrac{}{\mathsf{r} : e\backslash t}\,\text{Lex}}{\mathsf{r}\,\mathsf{i} : t}\,\backslash}{\mathsf{r}\,\mathsf{i} : \langle t\rangle}\,\text{Eval}}{\dfrac{\dfrac{\mathsf{r}\,\mathsf{i} : t}{(\mathsf{r}\,\mathsf{i})^{\eta} : \mathsf{M}t}\,\eta}{\lambda k.\,k\,(\mathsf{r}\,\mathsf{i})^{\eta} : \mathsf{K}_t\mathsf{M}t}\,\text{FA.1}}\,/$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(\multimap) : \mathsf{K}_t t/\mathsf{M}t}\,\text{Ax}}{\lambda k.\,k\,(\multimap) : \mathsf{K}_t(\mathsf{K}_t t/\mathsf{M}t)}\,\text{FA.1} \quad \dfrac{\dfrac{\lambda k.\,\mathbf{a.rel}_x \multimap k\,(\mathbf{if}\,(\mathsf{d}\,x)^{\eta}) : \mathsf{K}_t(\mathsf{M}t/\mathsf{M}t) \quad \lambda k.\,k\,(\mathsf{r}\,\mathsf{i})^{\eta} : \mathsf{K}_t\mathsf{M}t}{\lambda k.\,\mathbf{a.rel}_x \multimap k\,(\mathbf{if}\,(\mathsf{d}\,x)^{\eta}(\mathsf{r}\,\mathsf{i})^{\eta}) : \mathsf{K}_t\mathsf{M}t}\,\mathbf{S}_{/}}{\lambda k.\,\mathbf{a.rel}_x \multimap k\,(\mathsf{d}\,x \Rightarrow \mathsf{r}\,\mathsf{i})^{\eta} : \mathsf{K}_t\mathsf{M}t}\,\mathbf{if}}{\dfrac{\dfrac{\lambda k.\,\mathbf{a.rel}_x \multimap k\,(\lambda c.\,(\mathsf{d}\,x \Rightarrow \mathsf{r}\,\mathsf{i})^{\eta}_p \multimap c\,p) : \mathsf{K}_t\mathsf{K}_t t}{\lambda k.\,\mathbf{a.rel}_x \multimap k\,(\lambda c.\,c\,(\mathsf{d}\,x \Rightarrow \mathsf{r}\,\mathsf{i})) : \mathsf{K}_t\mathsf{K}_t t}\,\text{LeftID}}{\mathbf{a.rel}_x \multimap (\mathsf{d}\,x \Rightarrow \mathsf{r}\,\mathsf{i})^{\eta} : \mathsf{M}t}\,\text{FA.6}}\,\mathbf{S}_{/}$$

□

Fact A.13. Deriving ⟨a persuasive lawyer visits a relative of mine⟩ with differentiated indefinites.

Proof.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{a.law} : \mathsf{M}e}\,\text{Lex}}{\lambda k.\,\mathbf{a.law}_x \multimap k\,x : \mathsf{K}_t e}\multimap}{\lambda c.\,c\,(\lambda k.\,\mathbf{a.law}_x \multimap k\,x) : \mathsf{K}_{\mathsf{M}t}(\mathsf{K}_t e)}\,\text{FA.1} \quad \dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{v} : (e\backslash t)/e}\,\text{Lex}}{\lambda k.\,k\,\mathsf{v} : \mathsf{K}_t((e\backslash t)/e)}\,\text{FA.1}}{\lambda c.\,c\,(\lambda k.\,k\,\mathsf{v}) : \mathsf{K}_{\mathsf{M}t}(\mathsf{K}_t((e\backslash t)/e))}\,\text{FA.1} \quad \dfrac{\dfrac{\dfrac{}{\mathbf{a.rel} : \mathsf{M}e}\,\text{Lex}}{\lambda k.\,\mathbf{a.rel}_y \multimap k\,y : \mathsf{K}_t e}\multimap}{\lambda c.\,\mathbf{a.rel}_y \multimap c\,(\lambda k.\,k\,y) : \mathsf{K}_{\mathsf{M}t}(\mathsf{K}_t e)}\,\text{FA.2}}{\lambda c.\,\mathbf{a.rel}_y \multimap c\,(\lambda k.\,k\,(\mathsf{v}\,y)) : \mathsf{K}_{\mathsf{M}t}(\mathsf{K}_{\mathsf{M}t}(e\backslash t))}\,\text{FA.3}}{\lambda c.\,\mathbf{a.rel}_y \multimap c\,(\lambda k.\,\mathbf{a.law}_x \multimap k\,(\mathsf{v}\,y\,x)) : \mathsf{K}_{\mathsf{M}t}(\mathsf{K}_t t)}\,\text{FA.3}$$

$$\dfrac{\dfrac{\dfrac{}{\blacksquare : \mathsf{MM}t/\langle\mathsf{MM}t\rangle}\,\text{Lex} \quad \dfrac{\dfrac{\dfrac{\lambda c.\,\mathbf{a.rel}_y \multimap c\,(\lambda k.\,\mathbf{a.law}_x \multimap k\,(\mathsf{v}\,y\,x)) : \mathsf{K}_{\mathsf{M}t}(\mathsf{K}_t t)}{\lambda k.\,\mathbf{a.rel}_y \multimap k\,(\mathbf{a.law}_x \multimap (\mathsf{v}\,y\,x)^{\eta}) : \mathsf{K}_{\mathsf{M}t}\mathsf{M}t}\,\text{FA.5}}{\mathbf{a.rel}_y \multimap (\mathbf{a.law}_x \multimap (\mathsf{v}\,y\,x)^{\eta})^{\eta} : \mathsf{MM}t}\,\text{FA.4}}{\mathbf{a.rel}_y \multimap (\mathbf{a.law}_x \multimap (\mathsf{v}\,y\,x)^{\eta})^{\eta} : \langle\mathsf{MM}t\rangle}\,\text{Eval}}{\dfrac{\mathbf{a.rel}_y \multimap (\mathbf{a.law}_x \multimap (\mathsf{v}\,y\,x)^{\eta})^{\eta} : \mathsf{MM}t}{\lambda k.\,\mathbf{a.rel}_y \multimap k\,(\mathbf{a.law}_x \multimap (\mathsf{v}\,y\,x)^{\eta}) : \mathsf{K}_t\mathsf{M}t}\multimap}\,/$$

□

Note. The toy type system's coarse enough to let a string of the form "fin Γ and fin Δ" be parsed as fin · (Γ · (and · (fin · Δ))), in which case evaluation of Γ is not actually forced. Though this is somewhat peripheral to my main concerns—namely, what's possible *once obligatory evaluation transpires*—there's a number of options for enforcing a tighter connection between a *fin* and its scope island, including decomposing *fin* into two pieces which flank the island and elaborating the type system to force a tighter link between finite anchoring and tense (cf. Rizzi's 1997 Fin_0 and T_0).

A.7 The Reader.Set monad

Definition A.16 (The Reader.Set monad).

$$\begin{aligned} \mathsf{M}\,\alpha &::= s \to \alpha \to t \\ a^{\eta} &:= \lambda s.\, \{a\} \\ m_{v} \multimap \pi &:= \lambda s. \bigcup_{a \in m\,s} \pi[a/v]\, s \end{aligned}$$

Definition A.17 (Reader.Set lexicon sampling).

Item	**Meaning**	**Type**
John	j	e
saw	saw	$(e\backslash t)/e$
and	$(\wedge)$	$(t\backslash t)/t$
a linguist	$\textbf{a.ling} := \lambda s.\, \{x \mid \mathsf{ling}\, x\}$	$\mathsf{M}e$
she, GAP	$\textbf{pro} := \lambda s.\, \{s_{\top}\}$	$\mathsf{M}e$
not	$\textbf{not} := \lambda m s.\, \{\neg\exists a.\, a \in m\, s \wedge a\}$	$\mathsf{M}t/\mathsf{M}t$
if	$\textbf{if} := \lambda m n.\, \textbf{not}\, (m_p \multimap (\textbf{not}\, n)_q \multimap (p \wedge q)^{\eta})$	$(\mathsf{M}t/\mathsf{M}t)/\mathsf{M}t$
every linguist	$\textbf{ev.ling} := \lambda k.\, \textbf{not}\, ((\textbf{a.ling})_x \multimap \textbf{not}\, (k\, x))$	$\mathsf{K}_t\, e$
no linguist	$\textbf{no.ling} := \lambda k.\, \textbf{not}\, ((\textbf{a.ling})_x \multimap k\, x)$	$\mathsf{K}_t\, e$
or	$\textbf{or} := \lambda m n k s.\, m\, k\, s \cup n\, k\, s$	$(\mathsf{K}_R T\backslash \mathsf{K}_R T)/\mathsf{K}_R T$

Definition A.18 (Reader.Set dref introduction).

$$\frac{\Gamma \vdash E : \mathsf{K}_R T}{\Gamma \vdash \lambda k.\, E\, (\lambda a s.\, k\, a\, \widehat{sa}) : \mathsf{K}_R T}\,\blacktriangleright \qquad \frac{\Gamma \vdash E[\lambda s.\, (\textbf{pro}_v \multimap \pi)\, \widehat{sa}] : T}{\Gamma \vdash E[\lambda s.\, \pi[a/v]\, \widehat{sa}] : T}\,\text{Binding}$$

Note. Reader.Set's ► rule is equivalent to State.Set's ► rule, modulo types.

Fact A.14. Surface scope for *a man met every linguist*.

Proof. Same form as for FA.8. □

Fact A.15. Inverse scope for *a man met every linguist*.

Proof. Same form as for FA.9. □

Fact A.16. Indefiniteness scopes out of island on surface scope derivation. Nothing scopes out of island on inverse scope derivation.

Proof. Same form as for FA.10. □

Fact A.17. Deriving *a* > *if* reading of *if* ⟨*a relative of mine dies*⟩, ⟨*I'll be rich*⟩.

Proof. Same form as for FA.12. □

Fact A.18. Deriving ⟨a persuasive lawyer visits a relative of mine⟩ with differentiated indefinites:

Proof. Same form as for FA.13. □

Fact A.19. Deriving ⟨a linguist$_i$ rubbed his$_i$ head⟩.

Proof.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{a.ling} : \mathsf{M}e}\,\text{Lex}}{\lambda k.\,\mathbf{a.ling}_x \multimap k\,x : \mathsf{K}_t e}\multimap}{\lambda k.\,\mathbf{a.ling}_x \multimap \lambda s.\,k\,x\,\widehat{sx} : \mathsf{K}_t e}\blacktriangleright \quad \dfrac{\dfrac{\dfrac{}{\mathsf{r} : (e\backslash t)/e}\,\text{Lex}}{\lambda k.\,k\,\mathsf{r} : \mathsf{K}_t((e\backslash t)/e)}\,\text{FA.1} \quad \dfrac{\dfrac{\dfrac{}{\mathbf{pro} : \mathsf{M}e}\,\text{Lex}}{\lambda k.\,\mathbf{pro}_y \multimap k\,y : \mathsf{K}_t e}\multimap \quad \dfrac{\dfrac{}{\mathsf{h} : e\backslash e}\,\text{Lex}}{\lambda k.\,k\,\mathsf{h} : \mathsf{K}_t(e\backslash e)}}{\lambda k.\,\mathbf{pro}_y \multimap k\,(\mathsf{h}\,y) : \mathsf{K}_t e}\,\mathbf{S}_\backslash}{\lambda k.\,\mathbf{pro}_y \multimap k\,(\mathsf{r}\,(\mathsf{h}\,y)) : \mathsf{K}_t(e\backslash t)}\,\mathbf{S}_/}{\lambda k.\,\mathbf{a.ling}_x \multimap \lambda s.\,(\mathbf{pro}_y \multimap k\,(\mathsf{r}\,(\mathsf{h}\,y)\,x))\,\widehat{sx} : \mathsf{K}_t t}\,\mathbf{S}_\backslash}{\lambda k.\,\mathbf{a.ling}_x \multimap \lambda s.\,k\,(\mathsf{r}\,(\mathsf{h}\,x)\,x)\,\widehat{sx} : \mathsf{K}_t t}\,\text{Binding}}{\mathbf{a.ling}_x \multimap \lambda s.\,(\mathsf{r}\,(\mathsf{h}\,x)\,x)^\eta\,\widehat{sx} : \mathsf{M}t}\,\text{FA.4}}{\mathbf{a.ling}_x \multimap \lambda s.\,(\lambda s.\,\mathsf{r}\,(\mathsf{h}\,x)\,x)\,\widehat{sx} : \mathsf{M}t}\,\text{DA.16}}{\mathbf{a.ling}_x \multimap \lambda s.\,\mathsf{r}\,(\mathsf{h}\,x)\,x : \mathsf{M}t}\,\lambda}{\mathbf{a.ling}_x \multimap (\mathsf{r}\,(\mathsf{h}\,x)\,x)^\eta : \mathsf{M}t}\,\text{DA.16}}{\dfrac{}{\blacksquare : \mathsf{M}t/\langle \mathsf{M}t\rangle}\,\text{Lex} \quad \mathbf{a.ling}_x \multimap (\mathsf{r}\,(\mathsf{h}\,x)\,x)^\eta : \langle \mathsf{M}t\rangle}\,\text{Eval}}{\mathbf{a.ling}_x \multimap (\mathsf{r}\,(\mathsf{h}\,x)\,x)^\eta : \mathsf{M}t}\,/}{\lambda k.\,\mathbf{a.ling}_x \multimap k\,(\mathsf{r}\,(\mathsf{h}\,x)\,x) : \mathsf{K}_t t}\multimap$$

□

Note. Like the State.Set monad, indefiniteness scopes out of islands. Unlike the State.Set monad, post-evaluation binding is impossible: the dref-hosting $\widehat{sx}$ dies with evaluation.

A.8 Generalized combination

Definition A.19 (Generalizing the scopal type constructor).

$$\mathsf{K}_R^S T ::= S/(\mathsf{M}R/T) \qquad \mathsf{K}_R T ::= \mathsf{K}_R^{\mathsf{M}R} T$$

Definition A.20 (Generalized scopal combinators).

$$\dfrac{}{\varepsilon \vdash \mathbf{S}_\backslash : (\mathsf{K}_B^A T/\mathsf{K}_B^R(S\backslash T))/\mathsf{K}_R^A S}\,\text{Ax} \qquad \mathbf{S}_\backslash E\,F := \lambda k.\,E\,(\lambda x.\,F\,(\lambda f.\,k\,(f\,x)))$$

$$\dfrac{}{\varepsilon \vdash \mathbf{S}_/ : (\mathsf{K}_B^A T/\mathsf{K}_B^R S)/\mathsf{K}_R^A(T/S)}\,\text{Ax} \qquad \mathbf{S}_/ F\,E := \lambda k.\,F\,(\lambda f.\,E\,(\lambda x.\,k\,(f\,x)))$$

Note. Generalized versions of Facts A.1, A.2, A.3, A.4, A.5, and A.6 still hold.

A.9 Focus

Definition A.21 (The Focus monad).

$$\begin{aligned} \mathsf{M}\alpha &::= \alpha \times (\alpha \to t) \\ a^{\eta} &:= \langle a, \{a\} \rangle \\ m_v \multimap \pi &:= \langle \pi[m_0/v]_0, \bigcup_{a \in m_1} \pi[a/v]_1 \rangle \end{aligned}$$

Definition A.22 (Focus monad meaning sampling).

Item	**Meaning**	**Type**
F	$\mathbf{F} := \lambda a.\, \langle a, \mathsf{Alt}\, a \rangle$	$\alpha \to \mathsf{M}\alpha$
only	$\textbf{only} = \lambda \mathcal{P} x w.\, \{P \mid P \in \mathcal{P}_1 \wedge P\, x\, w\} = \mathcal{P}_0$	$\mathsf{M}(e \to t) \to e \to t$
also	$\textbf{also} = \lambda \mathcal{P} x w.\, \{P \mid P \in \mathcal{P}_1 \wedge P\, x\, w\} \supset \mathcal{P}_0$	$\mathsf{M}(e \to t) \to e \to t$

Fact A.20. Focus-monadic side effects scope out of islands.

Proof. See FA.7. □

Fact A.21. Focus-monadic side effects can be differentiated outside islands.

Proof. Analogous to FA.13. □

Appendix B

Monad law proofs

B.1 The monad laws

Definition.

$$\begin{aligned} a^{\eta} \multimap k &= k\,a && \text{LeftID} \\ m \multimap \eta &= m && \text{RightID} \\ (m \multimap k) \multimap c &= m \multimap \lambda a.\, k\,a \multimap c && \text{Assoc} \end{aligned}$$

B.2 Identity monad

Definition.

$$\begin{aligned} \mathsf{M}\alpha &::= \alpha \\ a^{\eta} &:= a \\ m \multimap k &:= k\,a \end{aligned}$$

LeftID holds.

$$\begin{aligned} a^{\eta} \multimap k &= a \multimap k \\ &= k\,a \end{aligned}$$

RightID holds.

$$\begin{aligned} m \multimap \eta &= \eta\,m \\ &= m \end{aligned}$$

Assoc holds.

$$\begin{aligned} (m \multimap k) \multimap c &= k\,m \multimap c \\ &= c\,(k\,m) \\ m \multimap \lambda a.\, k\,a \multimap c &= m \multimap \lambda a.\, c\,(k\,a) \\ &= c\,(k\,m) \end{aligned}$$

B.3 Reader monad

Definition.

$$\begin{aligned} \mathsf{M}\alpha &::= s \to \alpha \\ a^{\eta} &:= \lambda s.\, a \\ m \multimap k &:= \lambda s. k\,(m\,s)\,s \end{aligned}$$

LeftID holds.

$$\begin{aligned} a^{\eta} \multimap k &= (\lambda s.\, a) \multimap k \\ &= \lambda s.\, k\,a\,s \\ &= k\,a \end{aligned}$$

RightID holds.

$$\begin{aligned} m \multimap \eta &= m \multimap \lambda a s.\, a \\ &= \lambda s.\, m\,s \\ &= m \end{aligned}$$

Assoc holds.

$$\begin{aligned} (m \multimap k) \multimap c &= (\lambda s.\, k\,(m\,s)\,s) \multimap c \\ &= \lambda s.\, c\,(k\,(m\,s)\,s)\,s \\ m \multimap \lambda a.\, k\,a \multimap c &= m \multimap \lambda a s.\, c\,(k\,a\,s)\,s \\ &= \lambda s.\, c\,(k\,(m\,s)\,s)\,s \end{aligned}$$

B.4 Set monad

Definition.

$$\begin{aligned} \mathsf{M}\alpha &::= \alpha \to t \\ a^{\eta} &:= \{a\} \\ m \multimap k &:= \bigcup\{k\,a \mid a \in m\} \end{aligned}$$

LeftID holds.

$$\begin{aligned} a^{\eta} \multimap k &= \{a\} \multimap k \\ &= \bigcup\{k\,x \mid x \in \{a\}\} \\ &= k\,a \end{aligned}$$

RightID holds.

$$\begin{aligned} m \multimap \eta &= m \multimap \lambda a.\, \{a\} \\ &= \bigcup\{\{a\} \mid a \in m\} \\ &= m \end{aligned}$$

Assoc holds.

$$\begin{aligned}
(m \multimap k) \multimap c &= \bigcup\{k\,x \mid x \in m\} \multimap c \\
&= \bigcup\{c\,y \mid y \in \bigcup\{k\,x \mid x \in m\}\} \\
&= \bigcup\{c\,y \mid x \in m \wedge y \in k\,x\} \\
m \multimap \lambda a.\, k\,a \multimap c &= m \multimap \lambda a.\, \bigcup\{c\,y \mid y \in k\,a\} \\
&= \bigcup\{\bigcup\{c\,y \mid y \in k\,x\} \mid x \in m\} \\
&= \bigcup\{c\,y \mid x \in m \wedge y \in k\,x\}
\end{aligned}$$

B.5 Reader.Set monad

Definition.

$$\begin{aligned}
\mathsf{M}\alpha &::= s \to \alpha \to t \\
a^{\eta} &:= \lambda s.\, \{a\} \\
m \multimap k &:= \lambda s.\, \bigcup\{k\,a\,s \mid a \in m\,s\}
\end{aligned}$$

LeftID holds.

$$\begin{aligned}
a^{\eta} \multimap k &= (\lambda s.\, \{a\}) \multimap k \\
&= \lambda s.\, \bigcup\{k\,x\,s \mid x \in \{a\}\} \\
&= \lambda s.\, k\,a\,s \\
&= k\,a
\end{aligned}$$

RightID holds.

$$\begin{aligned}
m \multimap \eta &= m \multimap \lambda a s.\, \{a\} \\
&= \lambda s.\, \bigcup\{\{x\} \mid x \in ms\} \\
&= \lambda s.\, m\,s \\
&= m
\end{aligned}$$

Assoc holds.

$$\begin{aligned}
(m \multimap k) \multimap c &= (\lambda s.\, \bigcup\{k\,x\,s \mid x \in m\,s\}) \multimap c \\
&= \lambda s.\, \bigcup\{c\,y\,s \mid y \in \bigcup\{k\,x\,s \mid x \in m\,s\}\} \\
&= \lambda s.\, \bigcup\{c\,y\,s \mid x \in m\,s \wedge y \in k\,x\,s\} \\
m \multimap \lambda a.\, k\,a \multimap c &= m \multimap \lambda a s.\, \bigcup\{c\,y\,s \mid y \in k\,a\,s\} \\
&= \lambda s.\, \bigcup\{\bigcup\{c\,y\,s \mid y \in k\,x\,s\} \mid x \in m\,s\} \\
&= \lambda s.\, \bigcup\{c\,y\,s \mid x \in m\,s \wedge y \in k\,x\,s\}
\end{aligned}$$

B.6 State monad

Definition.

$$\begin{aligned} \mathsf{M}\alpha &::= s \to \alpha \times s \\ a^{\eta} &:= \lambda s.\, \langle a,\ s\rangle \\ m \multimap k &:= \lambda s.\, k\,(m\,s)_0\,(m\,s)_1 \end{aligned}$$

LeftID holds.

$$\begin{aligned} a^{\eta} \multimap k &= \lambda s.\, \langle a,\ s\rangle \multimap k \\ &= \lambda s.\, k\,a\,s \\ &= k\,a \end{aligned}$$

RightID holds.

$$\begin{aligned} m \multimap \eta &= m \multimap \lambda a s.\, \langle a,\ s\rangle \\ &= \lambda s.\, \langle (m\,s)_0,\ (m\,s)_1\rangle \\ &= \lambda s.\, m\,s \\ &= m \end{aligned}$$

Assoc holds.

$$\begin{aligned} (m \multimap k) \multimap c &= (\lambda s.\, k\,(m\,s)_0\,(m\,s)_1) \multimap c \\ &= \lambda s.\, c\,(k\,(m\,s)_0\,(m\,s)_1)_0\,(k\,(m\,s)_0\,(m\,s)_1)_1 \\ m \multimap \lambda a.\, k\,a \multimap c &= m \multimap \lambda a s.\, c\,(k\,a\,s)_0\,(k\,a\,s)_1 \\ &= \lambda s.\, c\,(k\,(m\,s)_0\,(m\,s)_1)_0\,(k\,(m\,s)_0\,(m\,s)_1)_1 \end{aligned}$$

B.7 State.Set monad

Definition.

$$\begin{aligned} \mathsf{M}\alpha &::= s \to \alpha \times s \to t \\ a^{\eta} &:= \lambda s.\, \{\langle a,\ s\rangle\} \\ m \multimap k &:= \lambda s.\, \bigcup\{k\,a\,s' \mid \langle a,\ s'\rangle \in m\,s\} \end{aligned}$$

LeftID holds.

$$\begin{aligned} a^{\eta} \multimap k &= \lambda s.\, \{\langle a,\ s\rangle\} \multimap k \\ &= \lambda s.\, \bigcup\{k\,x\,s' \mid \langle x, s'\rangle \in \{\langle a, s\rangle\}\} \\ &= \lambda s.\, k\,a\,s \\ &= k\,a \end{aligned}$$

RightID holds.

$$\begin{aligned} m \multimap \eta &= m \multimap \lambda a s.\, \{\langle a,\ s\rangle\} \\ &= \lambda s.\, \bigcup\{\{\langle x,\ s'\rangle\} \mid \langle x, s'\rangle \in m\, s\} \\ &= \lambda s.\, m\, s \\ &= m \end{aligned}$$

Assoc holds.

$$\begin{aligned} (m \multimap k) \multimap c &= (\lambda s.\, \bigcup\{k\, x\, s' \mid \langle x,\ s'\rangle \in m\, s\}) \multimap c \\ &= \lambda s.\, \bigcup\{c\, y\, s'' \mid \langle y,\ s''\rangle \in \bigcup\{k\, x\, s' \mid \langle x,\ s'\rangle \in m\, s\}\} \\ &= \lambda s.\, \bigcup\{c\, y\, s'' \mid \langle x,\ s'\rangle \in m\, s \wedge \langle y,\ s''\rangle \in k\, x\, s'\} \\ m \multimap \lambda a.\, k\, a \multimap c &= m \multimap \lambda a s'.\, \bigcup\{c\, y\, s'' \mid \langle y,\ s''\rangle \in k\, a\, s'\} \\ &= \lambda s.\, \bigcup\{\bigcup\{c\, y\, s'' \mid \langle y,\ s''\rangle \in k\, x\, s'\} \mid \langle x,\ s'\rangle \in m\, s\} \\ &= \lambda s.\, \bigcup\{c\, y\, s'' \mid \langle x,\ s'\rangle \in m\, s \wedge \langle y,\ s''\rangle \in k\, x\, s'\} \end{aligned}$$

B.8 Continuation monad

Definition.

$$\begin{aligned} \mathsf{M}\alpha &::= (\alpha \to \varrho) \to \varrho \\ a^{\eta} &:= \lambda k.\, k\, a \\ m \multimap k &:= \lambda c.\, m\, (\lambda a.\, k\, a\, c) \end{aligned}$$

LeftID holds.

$$\begin{aligned} a^{\eta} \multimap k &= (\lambda k.\, k\, a) \multimap k \\ &= \lambda c.\, (\lambda k.\, k\, a)\, (\lambda x.\, k\, x\, c) \\ &= \lambda c.\, k\, a\, c \\ &= k\, a \end{aligned}$$

RightID holds.

$$\begin{aligned} m \multimap \eta &= m \multimap (\lambda a k.\, k\, a) \\ &= \lambda c.\, m\, (\lambda x.\, (\lambda a k.\, k\, a)\, x\, c) \\ &= \lambda c.\, m\, (\lambda x.\, c\, x) \\ &= \lambda c.\, m\, c \\ &= m \end{aligned}$$

Assoc holds.

$$\begin{aligned}
(m \multimap k) \multimap c &= (\lambda c.\, m\,(\lambda x.\, k\,x\,c)) \multimap c \\
&= \lambda \gamma.\, (\lambda c.\, m\,(\lambda x.\, k\,x\,c))\,(\lambda y.\, c\,y\,\gamma) \\
&= \lambda \gamma.\, m\,(\lambda x.\, k\,x\,(\lambda y.\, c\,y\,\gamma)) \\
m \multimap \lambda a.\, k\,a \multimap c &= m \multimap \lambda a \gamma.\, k\,a\,(\lambda y.\, c\,y\,\gamma) \\
&= \lambda \gamma.\, m\,(\lambda x.\, (\lambda a \gamma.\, k\,a\,(\lambda y.\, c\,y\,\gamma))\,x\,\gamma) \\
&= \lambda \gamma.\, m\,(\lambda x.\, k\,x\,(\lambda y.\, c\,y\,\gamma))
\end{aligned}$$

The Continuation monad$^+$ has a more general type constructor, but this doesn't affect the proofs of the monad laws.

B.9 Focus monad

Definition.

$$\begin{aligned}
\mathsf{M}\alpha &::= \alpha \times (\alpha \to t) \\
a^\eta &:= \langle a,\, \{a\} \rangle \\
m \multimap k &:= \langle (k\,m_0)_0,\, \bigcup\{(k\,a)_1 \mid a \in m_1\} \rangle
\end{aligned}$$

LeftID holds.

$$\begin{aligned}
a^\eta \multimap k &= \langle a,\, \{a\} \rangle \multimap k \\
&= \langle (k\,a)_0,\, \bigcup\{(k\,x)_1 \mid x \in \{a\}\} \rangle \\
&= \langle (k\,a)_0,\, (k\,a)_1 \rangle \\
&= k\,a
\end{aligned}$$

RightID holds.

$$\begin{aligned}
m \multimap \eta &= m \multimap \lambda a.\, \langle a,\, \{a\} \rangle \\
&= \langle \langle m_0,\, \{m_0\} \rangle_0,\, \bigcup\{\langle x,\, \{x\} \rangle_1 \mid x \in m_1\} \rangle \\
&= \langle m_0,\, \bigcup\{\{x\} \mid x \in m_1\} \rangle \\
&= \langle m_0,\, m_1 \rangle \\
&= m
\end{aligned}$$

Assoc holds.

$$\begin{aligned}
(\langle a,\, A \rangle \multimap k) \multimap c &= \langle (k\,a)_0,\, \bigcup\{(k\,x)_1 \mid x \in A\} \rangle \multimap c \\
&= \langle (c\,(k\,a)_0)_0,\, \bigcup\{(c\,y)_1 \mid y \in \bigcup\{(k\,x)_1 \mid x \in A\}\} \rangle \\
&= \langle (c\,(k\,a)_0)_0,\, \bigcup\{(c\,y)_1 \mid x \in A \wedge y \in (k\,x)_1\} \rangle \\
\langle a,\, A \rangle \multimap \lambda a.\, k\,a \multimap c &= \langle a,\, A \rangle \multimap \lambda a.\, \langle (c\,(k\,a)_0)_0,\, \bigcup\{(c\,y)_1 \mid y \in (k\,a)_1\} \rangle \\
&= \langle (c\,(k\,a)_0)_0,\, \bigcup\{\bigcup\{(c\,y)_1 \mid y \in (k\,x)_1\} \mid x \in A\} \rangle \\
&= \langle (c\,(k\,a)_0)_0,\, \bigcup\{(c\,y)_1 \mid x \in A \wedge y \in (k\,x)_1\} \rangle
\end{aligned}$$

Bibliography

Abbott, Barbara. 2002. Donkey Demonstratives. *Natural Language Semantics* 10(4). 285–298. http://dx.doi.org/10.1023/A:1022141232323.

Abusch, Dorit. 1994. The scope of indefinites. *Natural Language Semantics* 2(2). 83–135. http://dx.doi.org/10.1007/BF01250400.

Aloni, Maria. 2007. Free choice, modals, and imperatives. *Natural Language Semantics* 15(1). 65–94. http://dx.doi.org/10.1007/s11050-007-9010-2.

Alonso-Ovalle, Luis. 2006. *Disjunction in Alternative Semantics*: University of Massachusetts, Amherst Ph.D. thesis.

AnderBois, Scott. 2010. Sluicing as Anaphora to Issues. In Nan Li & David Lutz (eds.), *Proceedings of Semantics and Linguistic Theory 20*, 451–470. Ithaca, NY: Cornell University.

Bach, Emmon & Robin Cooper. 1978. The NP-S analysis of relative clauses and compositional semantics. *Linguistics and Philosophy* 2(1). 145–150. http://dx.doi.org/10.1007/BF00365132. http://dx.doi.org/10.1007/BF00365132.

Barker, Chris. 1995. *Possessive Descriptions*. Stanford: CSLI Publications. 1991 Ph.D. dissertation, University of California, Santa Cruz.

Barker, Chris. 2002. Continuations and the Nature of Quantification. *Natural Language Semantics* 10(3). 211–242. http://dx.doi.org/10.1023/A:1022183511876.

Barker, Chris. 2012. Quantificational Binding Does Not Require C-Command. *Linguistic Inquiry* 43(4). 614–633. http://dx.doi.org/doi:10.1162/ling_a_00108.

Barker, Chris. 2013. Scopability and sluicing. *Linguistics and Philosophy* 36(3). 187–223. http://dx.doi.org/10.1007/s10988-013-9137-1.

Barker, Chris & Chung-chieh Shan. 2006. Types as Graphs: Continuations in Type Logical Grammar. *Journal of Logic, Language and Information* 15(4). 331–370. http://dx.doi.org/10.1007/s10849-006-0541-6.

Barker, Chris & Chung-chieh Shan. 2008. Donkey Anaphora is In-Scope Binding. *Semantics & Pragmatics* 1(1). 1–46. http://dx.doi.org/10.3765/sp.1.1.

Barker, Chris & Chung-chieh Shan. 2014. *Continuations and Natural Language*. Oxford: Oxford University Press.

Barros, Matthew. 2013. Harmonic sluicing: Which remnant/correlate pairs work and why. In Todd Snider (ed.), *Proceedings of Semantics and Linguistic Theory 23*, 295–315.

Barss, Andrew & Howard Lasnik. 1986. A Note on Anaphora and Double Objects. *Linguistic Inquiry*

17(2). 347–354.

Barwise, Jon. 1987. Noun Phrases, Generalized Quantifiers, and Anaphora. In Peter Gärdenfors (ed.), *Generalized Quantifiers*, 1–29. Dordrecht: Reidel.

van den Berg, Martin H. 1996. *Some aspects of the internal structure of discourse. The dynamics of nominal anaphora*: University of Amsterdam Ph.D. thesis.

Bittner, Maria. 2001. Surface Composition as Bridging. *Journal of Semantics* 18(2). 127–177. http://dx.doi.org/10.1093/jos/18.2.127.

Bittner, Maria. 2014. *Temporality: Universals and Variation*. Malden, MA, Oxford: Wiley-Blackwell. http://dx.doi.org/10.1002/9781118584002.

Brasoveanu, Adrian. 2007. *Structured nominal and modal reference*: Rutgers University Ph.D. thesis.

Brasoveanu, Adrian. 2008. Donkey pluralities: plural information states versus non-atomic individuals. *Linguistics and Philosophy* 31(2). 129–209. http://dx.doi.org/10.1007/s10988-008-9035-0.

Brasoveanu, Adrian & Donka F. Farkas. 2011. How indefinites choose their scope. *Linguistics and Philosophy* 34(1). 1–55. http://dx.doi.org/10.1007/s10988-011-9092-7.

Brennan, Jonathan. 2011. Indefinites, Choice Functions, and Discourse Anaphora. In Suzi Lima, Kevin Mullin & Brian Smith (eds.), *NELS 39: Proceedings of the 39th Annual Meeting of the North East Linguistic Society*, 151–163. Amherst, MA: GLSA.

Bumford, Dylan. 2013. Universal quantification as iterated conjunction. In Maria Aloni, Michael Franke & Floris Roelofsen (eds.), *Logic, Language and Meaning. Proceedings of the 19th Amsterdam Colloquium*, 67–74.

Büring, Daniel. 2004. Crossover Situations. *Natural Language Semantics* 12(1). 23–62. http://dx.doi.org/10.1023/B:NALS.0000011144.81075.a8.

Büring, Daniel. 2005. *Binding Theory*. New York: Cambridge University Press.

Cable, Seth. 2010. *The Grammar of Q. Q-particles, Wh-movement, and Pied Piping*. Oxford University Press.

Charlow, Simon. 2008. Free and Bound Pro-Verbs: A Unified Treatment of Anaphora. In Tova Friedman & Satoshi Ito (eds.), *Proceedings of Semantics and Linguistic Theory 18*, 194–211.

Charlow, Simon. 2010. Two kinds of binding out of DP. Unpublished ms.

Charlow, Simon. 2011. Codetermination. Unpublished ms.

Charlow, Simon. 2012. Cross-Categorial Donkeys. In Maria Aloni, Vadim Kimmelman, Floris Roelofsen, Galit W. Sassoon, Katrin Schulz & Matthijs Westera (eds.), *Logic, Language and Meaning*, vol. 7218 Lecture Notes in Computer Science, 261–270. Springer Berlin Heidelberg. http://dx.doi.org/10.1007/978-3-642-31482-7_27.

Charlow, Simon. 2014. Kennedy's Puzzle and Direct Interpretation. Unpublished ms.

Chierchia, Gennaro. 2001. A Puzzle about Indefinites. In Carlo Cecchetto, Gennaro Chierchia & Maria Teresa Guasti (eds.), *Semantic Interfaces: Reference, Anaphora, and Aspect*, 51–89. Stanford: CSLI Publications.

Chierchia, Gennaro. 2005. Definites, Locality, and Intentional Identity. In Gregory N. Carlson & Francis Jeffry Pelletier (eds.), *Reference and Quantification: The Partee Effect*, 143–177.

Stanford: CSLI Publications.

Chomsky, Noam. 2008. On phases. In Robert Freidin, Carlos Otero & Maria-Luisa Zubizaretta (eds.), *Foundational Issues in Linguistic Theory*, 133–166. Cambridge, MA: MIT Press.

Chung, Sandra, William A. Ladusaw & James McCloskey. 1995. Sluicing and logical form. *Natural Language Semantics* 3(3). 239–282. http://dx.doi.org/10.1007/BF01248819.

Church, Alonzo. 1936. An Unsolvable Problem of Elementary Number Theory. *American Journal of Mathematics* 58(2). 345–363.

Constant, Noah. 2012. Witnessable quantifiers license type-e meaning: Evidence from contrastive topic, equatives and supplements. In Anca Chereches (ed.), *Proceedings of Semantics and Linguistic Theory 22*, 286–306. eLanguage.

Cooper, Robin. 1983. *Quantification and syntactic theory*. Dordrecht: Springer Science+Business Media.

Dalrymple, Mary, Stuart M. Shieber & Fernando C. N. Pereira. 1991. Ellipsis and higher-order unification. *Linguistics and Philosophy* 14(4). 399–452. http://dx.doi.org/10.1007/BF00630923.

Danvy, Olivier & Andrzej Filinski. 1990. Abstracting control. In *Proceedings of the 1990 ACM conference on Lisp and functional programming*, 151–160. New York: ACM Press.

Dayal, Veneeta. 1996. *Locality in Wh quantification*. Dordrecht: Springer Science+Business Media.

Dayal, Veneeta. 2002. Single-pair versus multiple-pair answers: Wh-in-situ and scope. *Linguistic Inquiry* 33(3). 512–520. http://dx.doi.org/10.1162/ling.2002.33.3.512.

Dekker, Paul. 1994. Predicate Logic with Anaphora. In Mandy Harvey & Lynn Santelmann (eds.), *Proceedings of Semantics and Linguistic Theory 4*, 79–95. Ithaca, NY: Cornell University.

van Eijck, Jan. 2001. Incremental Dynamics. *Journal of Logic, Language and Information* 10(3). 319–351. http://dx.doi.org/10.1023/A:1011251627260.

Elbourne, Paul. 2005. *Situations and Individuals*. Cambridge, MA: MIT Press.

Engdahl, Elisabet. 1986. *Constituent Questions*. Dordrecht: Reidel.

Evans, Frederic. 1988. Binding into Anaphoric Verb Phrases. In Joyce Powers & Kenneth de Jong (eds.), *Proceedings of the 5th Annual Eastern States Conference on Linguistics*, 122–129. ESCOL Publications Committee: Linguistics Department, The Ohio State University.

Farkas, Donka F. 1981. Quantifier scope and syntactic islands. In Roberta Hendrick, Carrie Masek & Mary Frances Miller (eds.), *Papers from the Seventh Regional Meeting, Chicago Linguistic Society*, 59–66. Chicago: CLS.

Fiengo, Robert & Robert May. 1994. *Indices and Identity*. Cambridge, MA: MIT Press.

Filinski, Andrzej. 1994. Representing Monads. In *Proceedings of the 21st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 446–457. New York: ACM Press.

von Fintel, Kai. 2000. Singleton Indefinites (re. Schwarzschild 2000). MIT Syntax-Semantics Reading Group Talk.

Fodor, Janet Dean. 1970. *The Linguistic Description of Opaque Contexts*: Massachusetts Institute of Technology Ph.D. thesis.

Fodor, Janet Dean & Ivan A. Sag. 1982. Referential and quantificational indefinites. *Linguistics and Philosophy* 5(3). 355–398. http://dx.doi.org/10.1007/BF00351459.

Fox, Danny. 1999. Focus, parallelism and accommodation. In Tanya Matthews & Devon Strolovitch (eds.), *Proceedings of Semantics and Linguistic Theory 19*, 70–90. Ithaca, NY: Cornell University.

Fox, Danny. 2012. Lectures on the semantics of questions. Unpublished lecture notes.

Frege, Gottlob. 1892. Über Begriff und Gegenstand. *Vierteljahresschrift für wissenschaftliche Philosophie* 16. 192–205.

Geurts, Bart. 2000. Indefinites and Choice Functions. *Linguistic Inquiry* 31(4). 731–738. http://dx.doi.org/10.1162/002438900554550.

Giorgolo, Gianluca & Ash Asudeh. 2012. $\langle M, \eta, \star \rangle$ Monads for Conventional Implicatures. In Ana Aguilar Guevara, Anna Chernilovskaya & Rick Nouwen (eds.), *Proceedings of Sinn und Bedeutung 16*, 265–278. MIT Working Papers in Linguistics.

Giorgolo, Gianluca & Christina Unger. 2009. Coreference without Discourse Referents. In Barbara Plank, Erik Tjong Kim Sang & Tim Van de Cruys (eds.), *Proceedings of the 19th Meeting of Computational Linguistics in the Netherlands*, 69–81.

Groenendijk, Jeroen & Floris Roelofsen. 2009. Inquisitive Semantics and Pragmatics. In Jesus M. Larrazabal & Laraitz Zubeldia (eds.), *Meaning, Content, and Argument: Proceedings of the ILCLI International Workshop on Semantics, Pragmatics, and Rhetoric*, Bilbao: University of the Basque Country Press.

Groenendijk, Jeroen & Martin Stokhof. 1990. Dynamic Montague Grammar. In Laszlo Kalman & Laszlo Polos (eds.), *Proceedings of the Second Symposium on Logic and Language*, 3–48. Budapest: Eötvös Loránd University Press.

Groenendijk, Jeroen & Martin Stokhof. 1991. Dynamic predicate logic. *Linguistics and Philosophy* 14(1). 39–100. http://dx.doi.org/10.1007/BF00628304.

de Groote, Philippe. 2001. Type raising, continuations, and classical logic. In Robert van Rooy & Martin Stokhof (eds.), *Proceedings of the Thirteenth Amsterdam Colloquium*, 97–101. University of Amsterdam.

de Groote, Philippe. 2006. Towards a Montagovian account of dynamics. In Masayuki Gibson & Jonathan Howell (eds.), *Proceedings of Semantics and Linguistic Theory 16*, 1–16. Ithaca, NY: Cornell University.

Grosz, Barbara J., Aravind K. Joshi & Scott Weinstein. 1995. Centering: A Framework for Modeling the Local Coherence of Discourse. *Computational Linguistics* 21(2). 203–225.

Hackl, Martin. 2000. *Comparative Quantifiers*: Massachusetts Institute of Technology Ph.D. thesis.

Hagstrom, Paul. 1998. *Decomposing Questions*: Massachusetts Institute of Technology Ph.D. thesis.

Hamblin, C. L. 1973. Questions in Montague English. *Foundations of Language* 10(1). 41–53.

Hardt, Daniel. 1993. *VP Ellipsis: Form, Meaning, and Processing*: University of Pennsylvania dissertation.

Hardt, Daniel. 1999. Dynamic Interpretation of Verb Phrase Ellipsis. *Linguistics and Philosophy* 22. 187–221.

Hardt, Daniel, Nicholas Asher & Julie Hunter. 2013. VP ellipsis without indices. In Todd Snider (ed.), *Proceedings of Semantics and Linguistic Theory 23*, 239–256. Ithaca, NY: Cornell University.

Hartman, Jeremy. 2011. The Semantic Uniformity of Traces: Evidence from Ellipsis Parallelism. *Linguistic Inquiry* 42(3). 367–388. http://dx.doi.org/10.1162/LING_a_00050.

Heim, Irene. 1982. *The Semantics of Definite and Indefinite Noun Phrases*: University of Massachusetts, Amherst Ph.D. thesis.

Heim, Irene. 1983. On the Projection Problem for Presuppositions. In Michael Barlow, Daniel P. Flickinger & Michael T. Wescoat (eds.), *Proceedings of the Second West Coast Conference on Formal Linguistics*, 114–125. Stanford: Stanford University Press.

Heim, Irene. 1990. E-Type pronouns and donkey anaphora. *Linguistics and Philosophy* 13(2). 137–177. http://dx.doi.org/10.1007/BF00630732.

Heim, Irene. 1997. Predicates or formulas? Evidence from ellipsis. In Aaron Lawson (ed.), *Proceedings of Semantics and Linguistic Theory 7*, 197–221. Ithaca, NY: Cornell University.

Heim, Irene. 1998. Anaphora and Semantic Interpretation: A Reinterpretation of Reinhart's Approach. In Uli Sauerland & Orin Percus (eds.), *The Interpretive Tract*, vol. 25 MIT Working Papers in Linguistics, 205–246. Cambridge, MA: MIT Working Papers in Linguistics.

Heim, Irene & Angelika Kratzer. 1998. *Semantics in generative grammar*. Oxford: Blackwell.

Hendriks, Herman. 1993. *Studied Flexibility: Categories and Types in Syntax and Semantics*: University of Amsterdam Ph.D. thesis.

Hintikka, Jaakko. 1973. *Logic, Language-Games and Information*. Oxford: Clarendon Press.

Hirschbühler, Paul. 1982. VP-Deletion and Across-the Board Quantifier Scope. In James Pustejovsky & Peter Sells (eds.), *Proceedings of the North East Linguistic Society 12*, 132–139. Amherst, MA: GLSA.

Ionin, Tania & Ora Matushansky. 2006. The Composition of Complex Cardinals. *Journal of Semantics* 23(4). 315–360. http://dx.doi.org/10.1093/jos/ffl006.

Jacobson, Pauline. 1992. Antecedent Contained Deletion in a Variable-Free Semantics. In Chris Barker & David Dowty (eds.), *Proceedings of Semantics and Linguistic Theory 2* OSU Working Papers in Linguistics 40, 193–213.

Jacobson, Pauline. 1999. Towards a Variable-Free Semantics. *Linguistics and Philosophy* 22. 117–184. http://dx.doi.org/10.1023/A:1005464228727.

Jacobson, Pauline. 2000a. Paycheck Pronouns, Bach-Peters Sentences, and Variable-Free Semantics. *Natural Language Semantics* 8(2). 77–155. http://dx.doi.org/10.1023/A:1026517717879.

Jacobson, Pauline. 2000b. Paychecks, Stress, and Variable-Free Semantics. In Brendan Jackson & Tanya Matthews (eds.), *Proceedings of Semantics and Linguistic Theory 10*, 65–82. Ithaca, NY: Cornell University.

Jacobson, Pauline. 2004. Kennedy's Puzzle: What I'm Named or Who I Am? In Kazuha Watanabe & Robert B. Young (eds.), *Proceedings of Semantics and Linguistic Theory 14*, 145–162. Ithaca, NY: Cornell University.

Kamp, Hans. 1981. A theory of truth and semantic interpretation. In Jeroen Groenendijk, Theo Janssen

& Martin Stokhof (eds.), *Formal Methods in the Study of Language*, 277–322. Mathematical Centre Amsterdam.

Kamp, Hans & Uwe Reyle. 1993. *From Discourse to Logic*. Dordrecht: Kluwer Academic Publishers.

Kanazawa, Makoto. 1994. Weak vs. strong readings of donkey sentences and monotonicity inference in a dynamic setting. *Linguistics and Philosophy* 17(2). 109–158. http://dx.doi.org/10.1007/BF00984775.

Karttunen, Lauri. 1976. Discourse referents. In James D. McCawley (ed.), *Syntax and Semantics, volume 7: Notes from the Linguistic Underground*, 363–385. New York: Academic Press.

Karttunen, Lauri. 1977. Syntax and semantics of questions. *Linguistics and Philosophy* 1(1). 3–44. http://dx.doi.org/10.1007/BF00351935.

Keenan, Edward L. 1971. Names, quantifiers, and the sloppy identity problem. *Research on Language & Social Interaction* 4(2). 211–232.

Kennedy, Chris. 1994. Argument Contained Ellipsis. Linguistic Research Center Report LRC 94-03. UC Santa Cruz.

Kennedy, Chris. 2014. Predicates *and* Formulas: Evidence from Ellipsis. In Luka Crnič & Uli Sauerland (eds.), *The Art and Craft of Semantics: A Festschrift for Irene Heim*, vol. 1, 253–277. MIT Working Papers in Linguistics 70.

Keshet, Ezra. 2008. *Good Intensions: Paving Two Roads to a Theory of the De re/De dicto Distinction*: Massachusetts Institute of Technology Ph.D. thesis.

Klein, Ewan & Ivan A. Sag. 1985. Type-driven translation. *Linguistics and Philosophy* 8(2). 163–201. http://dx.doi.org/10.1007/BF00632365.

Kratzer, Angelika. 1991. The representation of focus. In Arnim von Stechow & Dieter Wunderlich (eds.), *Semantics: An international handbook of contemporary research*, chap. 40, 825–834. Berlin: de Gruyter.

Kratzer, Angelika. 1998. Scope or pseudoscope? Are there wide scope indefinites? In Susan Rothstein (ed.), *Events and Grammar*, 163–196. Dordrecht: Kluwer Academic Publishers.

Kratzer, Angelika. 2003. Choice functions in context. Unpublished ms.

Kratzer, Angelika & Junko Shimoyama. 2002. Indeterminate Pronouns: The View from Japanese. In Yukio Otsu (ed.), *Proceedings of the Third Tokyo Conference on Psycholinguistics*, 1–25. Tokyo: Hituzi Syobo.

Krifka, Manfred. 1989. Nominal Reference, Temporal Constitution and Quantification in Event Semantics. In Renate Bartsch, Johan van Bentham & Peter van Emde Boas (eds.), *Semantics and Contextual Expression*, 75–115. Dordrecht: Foris.

Krifka, Manfred. 1991. A Compositional Semantics for Multiple Focus Constructions. In Steve Moore & Adam Wyner (eds.), *Proceedings of Semantics and Linguistic Theory 1*, 127–158. Ithaca, NY: Cornell University.

Krifka, Manfred. 2006. Association with Focus Phrases. In Valéria Molnár & Susanne Winkler (eds.), *The Architecture of Focus*, 105–136. Mouton de Gruyter.

Larson, Richard K. 1988. On the Double Object Construction. *Linguistic Inquiry* 19(3). 335–391.

Lasersohn, Peter. 1998. Generalized Distributivity Operators. *Linguistics and Philosophy* 21(1). 83–93. http://dx.doi.org/10.1023/A:1005317815339.

Lasnik, Howard. 2014. Multiple Sluicing in English? *Syntax* 17(1). 1–20. http://dx.doi.org/10.1111/synt.12009.

Leu, Thomas. 2005. Donkey pronouns: Void Descriptions? In Leah Bateman & Cherlon Ussery (eds.), *NELS 35: Proceedings of the 35th Annual Meeting of the North East Linguistic Society*, 379–390. Amherst, MA: GLSA.

Lewis, David. 1970. General semantics. *Synthese* 22(1-2). 18–67. http://dx.doi.org/10.1007/BF00413598.

Lewis, David. 1975. Adverbs of Quantification. In Edward L. Keenan (ed.), *Formal Semantics of Natural Language*, 3–15. Cambridge: Cambridge University Press.

Liang, Sheng, Paul Hudak & Mark Jones. 1995. Monad Transformers and Modular Interpreters. In *22nd ACM Symposium on Principles of Programming Languages (POPL '95)*, 333–343. ACM Press.

Link, Godehard. 1998. *Algebraic Semantics in Language and Philosophy*. Stanford: CSLI Publications.

Mascarenhas, Salvador. 2009. *Inquisitive semantics and logic*: University of Amsterdam M.Sc. thesis.

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

Mayr, Clemens. 2012. Focusing bound pronouns. *Natural Language Semantics* 20(3). 299–348. http://dx.doi.org/10.1007/s11050-012-9083-4.

Merchant, Jason. 2001. *The Syntax of Silence: Sluicing, Islands, and the Theory of Ellipsis*. Oxford: Oxford University Press.

Moggi, Eugenio. 1989. Computational lambda-calculus and monads. In *Proceedings of the Fourth Annual Symposium on Logic in computer science*, 14–23. Piscataway, NJ, USA: IEEE Press.

Montague, Richard. 1974. The proper treatment of quantification in ordinary English. In Richmond Thomason (ed.), *Formal Philosophy*, chap. 8, 247–270. New Haven: Yale University Press.

Murthy, Chetan R. 1992. Control operators, hierarchies, and pseudo-classical type systems: A-translation at work. In Olivier Danvy & Carolyn L. Talcott (eds.), *Proceedings of the ACM SIGPLAN Workshop on Continuations*, 49–72. Technical report STAN-CS-92-1426, Stanford University, San Francisco, California, June 1992.

Muskens, Reinhard. 1996. Combining Montague semantics and discourse representation. *Linguistics and Philosophy* 19(2). 143–186. http://dx.doi.org/10.1007/BF00635836.

Nouwen, Rick. 2003. Complement Anaphora and Interpretation. *Journal of Semantics* 20(1). 73–113. http://dx.doi.org/10.1093/jos/20.1.73.

Nouwen, Rick. 2007. On Dependent Pronouns and Dynamic Semantics. *Journal of Philosophical Logic* 36(2). 123–154. http://dx.doi.org/10.1007/s10992-006-9029-8.

Novel, Marc & Maribel Romero. 2010. Movement, Variables and Hamblin Alternatives. In Martin Prinzhorn, Viola Schmitt & Sarah Zobel (eds.), *Proceedings of Sinn und Bedeutung 14*, 322–338.

Parsons, Terence. 1990. *Events in the Semantics of English*. Cambridge, MA: MIT Press.

Partee, Barbara H. 1973. Some transformational extensions of Montague grammar. *Journal of Philosophical Logic* 2(4). 509–534.

Partee, Barbara H. 1986. Noun Phrase Interpretation and Type-shifting Principles. In Jeroen Groenendijk, Dick de Jongh & Martin Stokhof (eds.), *Studies in Discourse Representation Theory and the Theory of Generalized Quantifiers*, 115–143. Dordrecht: Foris.

Partee, Barbara H. & Mats Rooth. 1983. Generalized conjunction and type ambiguity. In Rainer Bäuerle, Christoph Schwarze & Arnim von Stechow (eds.), *Meaning, Use and Interpretation of Language*, 361–383. Berlin: Walter de Gruyter.

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

Reinhart, Tanya. 1983. *Anaphora and Semantic Interpretation*. Chicago: University of Chicago Press.

Reinhart, Tanya. 1997. Quantifier Scope: How labor is Divided Between QR and Choice Functions. *Linguistics and Philosophy* 20(4). 335–397. http://dx.doi.org/10.1023/A:1005349801431.

Reinhart, Tanya. 2006. *Interface Strategies. Optimal and Costly Computations*. Cambridge, MA: MIT Press.

Rizzi, Luigi. 1997. The Fine Structure of the Left Periphery. In Liliane Haegeman (ed.), *Elements of Grammar* Kluwer International Handbooks of Linguistics, 281–337. Springer Netherlands. http://dx.doi.org/10.1007/978-94-011-5420-8_7.

Roelofsen, Floris. 2013. Algebraic foundations for the semantic treatment of inquisitive content. *Synthese* 190(1). 79–102. http://dx.doi.org/10.1007/s11229-013-0282-4.

Rooth, Mats. 1985. *Association with Focus*: University of Massachusetts, Amherst Ph.D. thesis.

Rooth, Mats. 1987. Noun Phrase Interpretation in Montague Grammar, File Change Semantics, and Situation Semantics. In Peter Gärdenfors (ed.), *Generalized Quantifiers*, 237–269. Dordrecht: Reidel.

Rooth, Mats. 1992a. Ellipsis redundancy and reduction redundancy. In Steven Berman & Arild Hestvik (eds.), *Proceedings of the Stuttgart Workshop on Ellipsis,* no. 29 in Arbeitspapiere des SFB 340, Stuttgart: University of Stuttgart.

Rooth, Mats. 1992b. A theory of focus interpretation. *Natural Language Semantics* 1(1). 75–116. http://dx.doi.org/10.1007/BF02342617.

Rooth, Mats. 1996. Focus. In Shalom Lappin (ed.), *The Handbook of Contemporary Semantic Theory*, 271–298. Oxford: Blackwell.

Rooth, Mats & Barbara H. Partee. 1982. Conjunction, type ambiguity, and wide scope 'or'. In Daniel P. Flickinger, Marlys Macken & Nancy Wiegand (eds.), *Proceedings of the First West Coast Conference on Formal Linguistics*, 353–362. Stanford: Stanford Linguistics Association.

Ross, John. 1967. *Constraints on variables in syntax*. Cambridge, MA: Massachusetts Institute of Technology dissertation.

Ruys, Eddy. 1992. *The Scope of Indefinites*: OTS, University of Utrecht Ph.D. thesis.

Safir, Ken. 2004. *The Syntax of (In)dependence*. Cambridge, MA: MIT Press.

Sag, Ivan A. 1976. *Deletion and logical form*: Massachusetts Institute of Technology Ph.D. thesis.

Sauerland, Uli. 2004. The Interpretation of Traces. *Natural Language Semantics* 12(1). 63–127. http://dx.doi.org/10.1023/B:NALS.0000011201.91994.4f.

Schlenker, Philippe. 2006. Scopal Independence: A Note on Branching and Wide Scope Readings of Indefinites and Disjunctions. *Journal of Semantics* 23(3). 281–314. http://dx.doi.org/10.1093/jos/ffl005.

Schlenker, Philippe. 2009. Local Contexts. *Semantics & Pragmatics* 2(3). 1–78. http://dx.doi.org/10.3765/sp.2.3.

Schlenker, Philippe. 2011. Singular Pronouns with Split Antecedents. *Snippets* 23(5). 13–15.

Schlenker, Philippe. 2012. Informativity-based maximality conditions. *Snippets* 26(7). 18–19. http://dx.doi.org/10.7358/snip-2012-026-schl.

Schubert, Lenhart K. & Francis Jeffry Pelletier. 1989. Generically speaking. In Barbara H. Partee, Gennaro Chierchia & Raymond Turner (eds.), *Properties, types and meaning, volume ii: Semantic issues*, 193–268. Dordrecht: Kluwer Academic Publishers.

Schwarz, Bernhard. 2000. *Topics in Ellipsis*: University of Massachusetts, Amherst Ph.D. thesis.

Schwarz, Bernhard. 2001. Two kinds of long-distance indefinites. In Robert van Rooy & Martin Stokhof (eds.), *Proceedings of the Thirteenth Amsterdam Colloquium*, 192–197. University of Amsterdam.

Schwarzschild, Roger. 1996. *Pluralities*. Dordrecht: Kluwer Academic Publishers.

Schwarzschild, Roger. 2002. Singleton Indefinites. *Journal of Semantics* 19(3). 289–314. http://dx.doi.org/10.1093/jos/19.3.289.

Shan, Chung-chieh. 2001. A Variable-Free Dynamic Semantics. In Robert van Rooy & Martin Stokhof (eds.), *Proceedings of the Thirteenth Amsterdam Colloquium*, University of Amsterdam.

Shan, Chung-chieh. 2002. Monads for natural language semantics. In Kristina Striegnitz (ed.), *Proceedings of the ESSLLI 2001 Student Session*, 285–298.

Shan, Chung-chieh. 2004. Binding alongside Hamblin alternatives calls for variable-free semantics. In Kazuha Watanabe & Robert B. Young (eds.), *Proceedings of Semantics and Linguistic Theory 14*, 289–304. Ithaca, NY: Cornell University.

Shan, Chung-chieh. 2005. *Linguistic Side Effects*: Harvard University Ph.D. thesis.

Shan, Chung-chieh & Chris Barker. 2006. Explaining Crossover and Superiority as Left-to-right Evaluation. *Linguistics and Philosophy* 29(1). 91–134. http://dx.doi.org/10.1007/s10988-005-6580-7.

Shimoyama, Junko. 2006. Indeterminate Phrase Quantification in Japanese. *Natural Language Semantics* 14(2). 139–173. http://dx.doi.org/10.1007/s11050-006-0001-5.

Slade, Benjamin. 2011. *Formal and philological inquiries into the nature of interrogatives, indefinites, disjunction, and focus in Sinhala and other languages*: University of Illinois Ph.D. thesis.

Solomon, Michael. 2011. True Distributivity and the Functional Interpretation of Indefinites. Unpublished ms. http://semanticsarchive.net/Archive/zkxN2M4M/solomon-functional-indefinites.pdf.

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

Stojnic, Una, Matthew Stone & Ernest Lepore. 2013. Discourse and Logical Form. Unpublished ms.

Stone, Matthew. 1992. *Or* and anaphora. In Chris Barker & David Dowty (eds.), *Proceedings of Semantics and Linguistic Theory 2* OSU Working Papers in Linguistics 40, 367–385.

de Swart, Henriëtte. 1999. Indefinites between Predication and Reference. In Tanya Matthews & Devon Strolovitch (eds.), *Proceedings of Semantics and Linguistic Theory 9*, 273–297. Ithaca, NY: Cornell University.

Szabó, Zoltán Gendler. 2011. Bare Quantifiers. *Philosophical Review* 120(2). 247–283. http://dx.doi.org/10.1215/00318108-2010-029.

Szabolcsi, Anna. 1992. Combinatory Grammar and Projection from the Lexicon. In Ivan A. Sag & Anna Szabolcsi (eds.), *Lexical Matters*, 241–268. Stanford: CSLI Publications.

Szabolcsi, Anna. 1997. Strategies for Scope Taking. In Anna Szabolcsi (ed.), *Ways of Scope Taking*, chap. 4, 109–154. Dordrecht: Kluwer Academic Publishers.

Szabolcsi, Anna. 2003. Binding on the Fly: Cross-Sentential Anaphora in Variable-Free Semantics. In Geert-Jan M. Kruijff & Richard T. Oehrle (eds.), *Resource-Sensitivity, Binding and Anaphora*, 215–227. Dordrecht: Kluwer Academic Publishers.

Szabolcsi, Anna. 2010. *Quantification*. New York: Cambridge University Press.

Szabolcsi, Anna. 2011. Scope and binding. In Klaus von Heusinger, Claudia Maienborn & Paul Portner (eds.), *Semantics: An International Handbook of Natural Language Meaning*, vol. 33 (HSK 2), chap. 62, 1605–1641. Berlin: de Gruyter. http://dx.doi.org/10.1515/9783110255072.1605.

Takahashi, Shoichi & Danny Fox. 2005. MaxElide and the re-binding problem. In Effi Georgala & Jonathan Howell (eds.), *Proceedings of Semantics and Linguistic Theory 15*, 223–240. Ithaca, NY: Cornell University.

Tomioka, Satoshi. 1999. A sloppy identity puzzle. *Natural Language Semantics* 7(2). 217–241. http://dx.doi.org/10.1023/A:1008309217917.

Unger, Christina. 2012. Dynamic Semantics as Monadic Computation. In Manabu Okumura, Daisuke Bekki & Ken Satoh (eds.), *New Frontiers in Artificial Intelligence JSAI-isAI 2011*, vol. 7258 Lecture Notes in Artificial Intelligence, 68–81. Springer Berlin Heidelberg. http://dx.doi.org/10.1007/978-3-642-32090-3_7.

Väänänen, Juokko. 2007. *Dependence Logic: A New Approach to Independence Friendly Logic*. New York: Cambridge University Press.

Wadler, Philip. 1992. Comprehending monads. In *Mathematical Structures in Computer Science*, vol. 2 (special issue of selected papers from 6th Conference on Lisp and Functional Programming), 461–493. http://dx.doi.org/10.1145/91556.91592.

Wadler, Philip. 1994. Monads and composable continuations. *Lisp and Symbolic Computation* 7(1). 39–56. http://dx.doi.org/10.1007/BF01019944.

Wadler, Philip. 1995. Monads for functional programming. In Johan Jeuring & Erik Meijer (eds.), *Advanced Functional Programming*, vol. 925 Lecture Notes in Computer Science, 24–52. Springer Berlin Heidelberg. http://dx.doi.org/10.1007/3-540-59451-5_2.

Wescoat, Michael. 1989. Sloppy Readings with Embedded Antecedents. Unpublished ms.

Williams, Edwin S. 1977. Discourse and Logical Form. *Linguistic Inquiry* 8(1). 101–139.

Winter, Yoad. 1997. Choice Functions and the Scopal Semantics of Indefinites. *Linguistics and Philosophy* 20(4). 399–467. http://dx.doi.org/10.1023/A:1005354323136.

Winter, Yoad. 1998. *Flexible Boolean semantics*: Utrecht University Ph.D. thesis.

Wold, Dag E. 1996. Long Distance Selective Binding: The Case of Focus. In Teresa Galloway & Justin Spence (eds.), *Proceedings of Semantics and Linguistic Theory 6*, 311–328. Ithaca, NY: Cornell University.

Zimmermann, Thomas Ede. 2000. Free Choice Disjunction and Epistemic Possibility. *Natural Language Semantics* 8(4). 255–290. http://dx.doi.org/10.1023/A:1011255819284.