Dynamic semantics with static types

Dylan Bumford
UCLA

Simon Charlow
Rutgers

Abstract Semantic analyses of natural language typically rely on variables for the interpretation of binding relationships. This is true of standard static setups, where sentences might denote sets of variable assignments, as well as standard dynamic ones, where they might denote relations between (sets of) variable assignments. Several well-known alternative frameworks eschew object-language variables in favor of representing semantic dependencies as functional dependencies. This obviates assignments, and has the benefit that any expression's binding needs are discoverable directly from its type. But these popular variable-free approaches are limited to static, in-scope binding relationships, those in which dependents occur in the arguments of their binders. In this paper we develop a semantics that is variable-free in the same sense, but captures traditional notions of dynamic anaphora. We demonstrate the value of anaphoric type transparency with novel analyses of crossover, ellipsis, and sloppy/paycheck anaphora, and compare the new semantics, which introduces the notions of parameterized monads and lenses to the linguistics literature, with other list-based dynamic systems and other accounts of crossover.

1 Introduction

One of the chief aims of dynamics semantics is to explain how an indefinite in one clause can come to bind a pronoun in another (Kamp 1981, Heim 1982, Groenendijk & Stokhof 1991, Muskens 1996, and many others). Analyses in this tradition are often built around a procedural metaphor anchored by the idea of *sequential* semantic interpretation. First meet a farmer f; next meet a donkey d; now learn that f owns d; and so on. On occasion, it is even claimed that all of semantic composition can be understood as a morpheme-by-morpheme sequence of dynamic updates to an information state (Bittner 2001; see also related discussions in Brasoveanu & Dotlačil 2020, Bott & Sternefeld 2017).

The trouble is that any robust compositional semantics for natural language must contend with the fact that some expressions *take scope* over syntactic material that precedes and/or dominates them. When this happens, expressions that are low/rightward/late end up controlling what information is available to other expressions that are high/leftward/early. Allowing for this sort of inversion of control both subverts the procedural metaphor and neutralizes vaunted evaluation order asymmetries.

In this paper we attempt to have our dynamic cake and eat it too. We develop a directly compositional, variable-free dynamic semantics with the following features:

- Contexts are finite, type-heterogeneous lists (and later, trees), constructively generated in the course of semantic interpretation
- Pro-forms, including ellipses, are polymorphic update functions on contexts, with traditional numerical or alphabetical indices replaced by two (and later, three) combinators that can be used to build arbitrarily deep lenses into contexts
- Every expression's type reveals both the discourse referents that it pushes and the unbound anaphoric dependencies that it contains
- The type system is predicative; all expressions are simply-typed or parametrically polymorphic, with principal types guaranteed to be inferrable
- Dynamic sequencing is monadic, enabling standard integrations with combinatorial theories of scope, and fine control over evaluation order

We will discuss and motivate these properties in the following sections, but it will likely be helpful to go ahead and sketch what we have in mind. Consonant with much of the literature on dynamic semantics, we conceive of a semantic update as a nondeterministic function that yields values by transitioning from an input context to a set of potential output contexts. Let $\iota \overset{\tau}{\Rightarrow} o$ abbreviate the type of such a function, which given an input ι produces values τ paired with outputs o: that is, $\iota \overset{\tau}{\Rightarrow} o \equiv \iota \to \{\tau \times o\}$. Crucially we allow that the output contexts o may be of a different type than the input context ι. In fact, in general we assume the introduction of discourse referents will change the type of a discourse state by recording the type of the referent so introduced. For instance, assuming the name 'John' introduces **j** as a referent, we might have the denotation in (1a); this is a computation that transitions from a state i to a state $(\mathbf{j}, i)$, yeilding the value **j** for composition with the rest of the sentence, as in (1b). Since the name does not request any particular information from the discourse context, the incoming state may be of any type whatsoever. In particular, it may just as well be empty: $()$.

(1a) $\text{John}^{\triangleright} : \Gamma \overset{e}{\Rightarrow} (e, \Gamma)$
$[\![\text{John}^{\triangleright}]\!] = \lambda i.\, \{\langle \mathbf{j}, (\mathbf{j}, i)\rangle\}$

(1b) $\text{John}^{\triangleright}\text{arrived} : \Gamma \overset{t}{\Rightarrow} (e, \Gamma)$
$[\![\text{John}^{\triangleright}\ \text{arrived}]\!] = \lambda i.\, \{\langle \mathbf{arrive}\ \mathbf{j}, (\mathbf{j}, i)\rangle\}$

Inversely, a pronoun requires an antecedent somewhere in its input, but does not otherwise advance the discourse. On the contrary, we suggest that a pronoun uses up its referent, removing it from the discourse record. The denotation in (2a) for instance reads in a context that has at least one value, and pulls that value out of the context for local composition. Notice that the anaphoric dependency of the pronoun, signaled by its type $(e, \Gamma) \overset{e}{\Rightarrow} \ldots$, is inherited by the containing expression in (2b), signaled likewise by its type $(e, \Gamma) \overset{t}{\Rightarrow} \ldots$.

(2a) $\text{he}_0 : (e, \Gamma) \overset{e}{\Rightarrow} \Gamma$
$[\![\text{he}_0]\!] = \lambda(x, i).\, \{\langle x, i\rangle\}$

(2b) $\text{he}_0\ \text{whistled} : (e, \Gamma) \overset{t}{\Rightarrow} \Gamma$
$[\![\text{he}_0\ \text{whistled}]\!] = \lambda(x, i).\, \{\langle \mathbf{whistle}\ x, i\rangle\}$

Sequencing an expression that adds a referent with an expression that consumes one has the expected cancelling effect. The outgoing state of the former is guaranteed to match the incoming requirements of the latter, so the composite type is once again entirely parametric in its input.

(3) $\text{John}^{\triangleright}\ \text{arrived; he}_0\ \text{whistled} : \Gamma \overset{t}{\Rightarrow} \Gamma$
$[\![\text{John}^{\triangleright}\ \text{arrived; he}_0\ \text{whistled}]\!] = \lambda i.\, \{\langle \mathbf{arrive}\ \mathbf{j} \wedge \mathbf{whistle}\ \mathbf{j}, i\rangle\}$

The details of pronouns and indices are given in Sections 3.1 and 5, but the upshot is that pronouns may impose unboundedly deep requirements on input contexts. The sentence in (4), for instance, requires a context that has been added to at least twice, second-most recently with an entity. When this requirement is met by a preceding expression, the dependency is discharged (6); when it is not, the dependency persists (7).

(4) $\text{she}_1\ \text{whistled} : (\alpha, (e, \Gamma)) \overset{t}{\Rightarrow} (\alpha, \Gamma)$
$[\![\text{she}_1\ \text{whistled}]\!] = \lambda(x, (y, i)).\, \{\langle \mathbf{whistle}\ y, (x, i)\rangle\}$

(5) $\text{Mary}^{\triangleright}\ \text{saw John}^{\triangleright} : \Gamma \overset{t}{\Rightarrow} (e, (e, \Gamma))$
$[\![\text{Mary}^{\triangleright}\ \text{saw John}^{\triangleright}]\!] = \lambda i.\, \{\langle \mathbf{see}\ \mathbf{j}\ \mathbf{m}, (\mathbf{j}, (\mathbf{m}, i))\rangle\}$

(6) Mary$^{\triangleright}$ saw John$^{\triangleright}$; she$_1$ whistled : $\Gamma \overset{\mathsf{t}}{\Rightarrow} (\mathsf{e}, \Gamma)$
$[\![\text{Mary}^{\triangleright} \text{ saw John}^{\triangleright}\text{; she}_1 \text{ whistled}]\!] = \lambda i.\{\langle \mathbf{see\ j\ m} \wedge \mathbf{whistle\ j},\ (\mathbf{j}, i)\rangle\}$

(7) John$^{\triangleright}$ arrived; she$_1$ whistled : $(\mathsf{e}, \Gamma) \overset{\mathsf{t}}{\Rightarrow} (\mathsf{e}, \Gamma)$
$[\![\text{John}^{\triangleright} \text{ arrived; she}_1 \text{ whistled}]\!] = \lambda(y, i).\{\langle \mathbf{arrive\ j} \wedge \mathbf{whistle}\ y,\ (\mathbf{j}, i)\rangle\}$

This is the sense in which the dynamic semantics we propose is *statically typed*. A pronoun without an antecedent in the discourse record is simply ill-typed. For example, the empty context $() : ()$ cannot serve as input to (7); the sentence and the empty context plainly do not have compatible types, and this incompatibility can be detected "at compile time", before the meaning of the sentence is evaluated in any way. This contrasts with standard dynamic approaches to anaphora, wherein the absence of a referent is only detectable "at runtime", typically when a pronoun looks to a partial assignment function that happens to be missing its index. That is, you have to try and compute the meaning of the sentence — to *run* it, so to speak — in order to find out whether it is defined.

We do not wish to suggest that static typing is inherently preferable to the alternative (see Section 8.4 for further discussion of this point), but this paper identifies several cases where static typing, and the concomitant type-level transparency, result in improved empirical coverage over the alternatives. On a technical level, previous uses of static types to drive theories of anaphora have either been limited to static, in-scope binding (e.g., variable-free semantics), or required the use of dependent types (e.g., Martin 2016), for which the problem of type inference is undecidable. The fragment we spell out here requires a comparatively simple type theory with so-called Hindley-Milner polymorphism, in which type inference is decidable and practical, and every expression is guaranteed to have a single, maximally general, inferrable type (see Pierce 2002 for discussion of Hindley-Milner type theory and comparison with dependent types).

As mentioned above, anaphora is expected to be heterogeneous. A context may in principle contain a value of any type at any position, and a pro-form may request a value of any type at any position. As long as the two match, anaphora will succeed. Typical dynamic frameworks — and static ones, for that matter — are not well-equipped for this sort of cross-categorial reference, as we discuss in Section 2. On the other hand, variable free theories of anaphora, discussed in Section 3, handle heterogeneity without any fuss at all, but have rarely been applied to cases of genuine out-of-scope binding. We attempt to bring these traditions together in Sections 4 and 5 in a way that we hope seems familiar. Section 6 on crossover illustrates the value of having expressions disclose the needs of their unresolved pronouns in their types, and Section 7 puts type-heterogeneity to work, demonstrating how various sorts of intensional anaphora may be derived, including paycheck pronouns and sloppy ellipsis. As desired, the resulting fragment maintains a linear (and dynamic) theory of binding alongside a traditional non-linear (and island-sensitive) theory of scope.

2 Dynamic semantics

The simplest and most commonly used variants of dynamic semantics only allow anaphora to type e, without any type-theoretic residue of unbound pronouns.

Yet there is a clear need for generalized binding and discourse reference. Most obviously, extraclausal anaphora is commonplace with any type-theoretically atomic sort of object (times, worlds, degrees, events, etc.). Indeed the availability of such anaphora is sometimes taken to *diagnose* the linguistic utility of a primitive type in the first place (see Rett 2022 for a survey of arguments along these lines). We also take it to be uncontroversial that propositional anaphors like 'too' and response particles like 'yes' and 'no' require semantic tracking of propositional (and

therefore typically functional) discourse referents. Cross-categorial movement and reconstruction create further dependencies involving typically functional types, forcing even static analyses to recognize asisgnments that sort variables by type (Heim & Kratzer 1998).

More interestingly, focus- and givenness-marking are generally recognized as anaphoric processes. Rooth (1985), for instance, famously introduced pronouns into logical forms to account for the relationship between an expression containing a focused constituent and its unfocused discourse antecedent. Since expressions of any (possibly functional) type may contain foci, it follows that antecedents and pronouns may be of any type as well. Relatedly, ellipsis is often seen as a species of anaphora, with linguistic gaps re-invoking the meanings of earlier expressions. And since ellipsis antecedents may contain arbitrarily complex arrangements of quantifiers, the discourse referents they generate may have types of arbitrarily high orders.

This diversity of pro-reference requires technical care, but isn't especially problematic for traditional Tarskian semantic schemes, where the assignnments that value variables exist in theoretical limbo between the object language and the model. An assignment with countably many variables of type $\mathtt{e}$ is not much different from an assignment with countably many variables of each type, and a suite of type-sorted pronouns does not add any real theoretical complexity to a grammar that already contains higher-order quantification. But many dynamic frameworks, and some static ones, reify these assignments in their models, and characterize the denotations of expressions as functions from and to assignments. If these expressions are to have types, the assignments they depend on must have types too. But if these assignments are functions from variables to values, and the values can be any sort of model-theoretic object whatsoever, then there is no choice but to introduce a universal type $\top$ that covers the union of every domain. Then an assignment may be of type $\mathtt{Var} \to \top$.

The trouble really begins when the referents of pronouns are themselves pronominal in some respect. This is perhaps clearest in the case of sloppy verb phrase ellipsis, where the ellipsis antecedent contains a bound pronoun that is re-bound in the elliptical context. For instance, in (8), the local discourse context of the second clause determines that the elliptical gap should be understood to mean whatever 'called his_x mom' means. And then the same local context is used to determine that the pronominal dependency in that denotation should resolve to whatever x is locally assigned to.

(8) John^x [called his_x mom]^y; Bill^x did_y too

(9) John^x spent [his_x paycheck]^y; Bill^x saved it_y

Similar mechanics govern paycheck readings in which a pronoun is anteceded by a nominal form that again contains a pronoun, as in (9). Imagine that the antecedent expression 'his paycheck' in (9) denotes the function from any assignment g to the paycheck of whoever g assigns to x. If 'John' binds x, then this refers to John's paycheck. Intuitively it is this function that the pronoun in the second clause of (9) invokes, now in a context where 'Bill' has captured the binding of x. That is, the local discourse context of the second clause is an assignment g' that maps y to the function $\lambda g.\, g_x$. The pronoun then refers to the value that this function takes at the same local context g'. In other words, the denotation of 'it' here is $\lambda g.\, g_y\, g$, a function that takes any assignment g, extracts the anaphoric denotation stored at g_y and applies this assignment-dependent meaning to g.

The problem, as Muskens (1995) points out, is that there is no such function. Let $\mathtt{g}$ abbreviate the type of assignments, $\mathtt{Var} \to \top$, and let v be a variable of type $\mathtt{g} \to \mathtt{t}$. Then if there were a function $\mathbf{pro} = \lambda g.\, g_v\, g$, of the sort needed for paycheck anaphora, it would also be of type $\mathtt{g} \to \mathtt{t}$. Being a function, it should apply to any assignment g in the domain of $\mathtt{g}$. In particular, it should apply to

any assignment g^* such that $g^* v = \lambda g. \neg(\mathbf{pro}\, g)$. But then $\mathbf{pro}\, g^* = g^*_v\, g^* = \neg(\mathbf{pro}\, g^*)$, which is impossible.

Muskens (1995, 1996) himself opted to treat heterogeneous discourse states not as assignments at all, but as abstract points in the model, akin to possible worlds. Variables then assign referents to states, rather than the other way around. But the potential for paradox arises just the same (Muskens 1995: p. 152, fn. 10), and so variables are implicitly prohibited from ranging over intensional objects (nothing of type $\mathtt{g} \to \ldots$). The solution yields a consistent logic, but at the cost of intensional anaphora. Hardt (1999) relaxes this assumption, specifically aiming to analyze discourses like (8) and (9), but gestures toward further restrictions that might be imposed on the syntax of the metalanguage so as to prevent recursive definitions.

3 Variable-free semantics

Rather than relying on a single, imperial notion of context capable of storing any type of semantic object, *variable-free* theories of binding solve the formal problem of heterogeneous anaphora by eliminating the idea of an anaphoric context altogether. Trivially, with no variables in the object language, there is no need for assignments in the metalanguage — heterogeneous or otherwise. In fact, variable-free fragments tend to be directly compositional, making no essential use of a metalanguage at all.

Instead, anaphoric elements like gaps, ellipses, and pronouns give rise to denotations that take additional, ordinary functional arguments. Those additional arguments provide the values of the contained dependencies. For instance, a sentence containing a single nominal pronoun, like (10), might denote a function from possible antecedents for that pronoun (that is, individuals) to whatever truth value that sentence would have if the pronoun referred to that antecedent. A sentence missing a single verb phrase, like (11), might denote a function from antecedents for that ellipsis (that is, properties) to whatever truth value that sentence would have if the ellipsis were filled in with that property. Combining the two sentences, as in (12) ought then to denote a function from individuals (to resolve the pronoun) and properties (to resolve the ellipsis) to truth values.

(10) She left

(11) John did

(12) She did

Already we may note two distinctive features of the variable-free semantic program. First, and most distinctively, it is plainly impossible therein for grammatical constraints to refer to object-language variable names, because there aren't any. One immediate consequence of this is the necessity of non-syntactic explanations for Binding Theoretic and crossover effects, which we return to in Sections 4 and 6. Second, if an expression harbors any unresolved anaphoric elements, then the type (and/or category) of that expression will reveal the elements' presence and nature. As in the example above, a sentence with one unbound pronoun is of type $\mathtt{e} \to \mathtt{t}$; a sentence with two unbound pronouns is of type $\mathtt{e} \to \mathtt{e} \to \mathtt{t}$; a sentence with two unbound pronouns, NP ellipsis, and VP ellipsis will have type $\mathtt{e} \to \mathtt{e} \to (\mathtt{e} \to \mathtt{t}) \to (\mathtt{e} \to \mathtt{t}) \to \mathtt{t}$; etc.

The tradeoff in treating constituents containing anaphors as ordinary functions awaiting antecedent arguments is that semantic combination can become a much more delicate and complex affair. Not every verb phrase will denote a property, and not every noun phrase an entity, so the blunt hammer of Function Application can hardly be expected to mediate all semantic combinations. Jacobson (1999, 2014), for instance, defines several infinite families of recursive combinators

approximating function composition and argument duplication at arbitrary (and arbitrarily many) argument positions. In light of this, adding a pronoun or an additional argument between a binder and a bindee can mean re-thinking all the combinators in a derivation.

And then there is scope. Without object-language variables, there is no object-language abstraction, and so no way to create *in the syntax* the sorts of derived properties characteristic of nuclear scopes. We return to this issue in Section 4, where we present, and adopt, Shan & Barker's (2006) combinatorial solution to a robust theory of scope-taking. First, though, we'll introduce a system of (static) pronouns that we will argue maintains the type-theoretical benefits of the variable-free vision, but paves the way for a more uniform picture of composition.

3.1 Inductive indices

One way to frame the above discussion is that total, heterogeneous assignments encode contexts that are maximally large. Expressions are provided with all of the information they could possibly need, whether or not they actually need any contextual information at all. The technique is a classic generalization to the worst case, and without some care the cases can get so bad as to be paradoxical. Conversely, variable-free theories that model anaphors as identity functions are maximally small. Expressions are provided with exactly the information required to settle the referents of open dependencies, and no more. The technique is a flat refusal to generalize, such that every pronoun demands its own singular context. As a result, not only is the *type* of an expression divorced from its underlying syntactic category, but also the *number* of contextual inputs required for evaluation — the very adicity of the denotation — becomes unpredictable.

We do not wish to suggest that either of these approaches are formally or empirically untenable, but we would like to introduce a notion of dependency that splits the difference between the two. Expressions, we propose, accept contexts that contain *at least* enough information to value their unbound anaphors. Note that both sorts of theories above are simply-typed; the context required for every expression is fully determined by its components. With assignments, this is because the input to any semantic evaluation is of type $\mathtt{Var} \to \top$; for variable-free semantics, this is because the inputs to any semantic evaluation are of whatever types its anaphors are. In contrast, the proposal here is to let denotations be as *polymorphic* as possible in their inputs. The types in (14)–(18) illustrate the difference between the three approaches.

(14) **variable-full**

she_0 left : $(\mathtt{Var} \to \top) \to \mathtt{t}$

she_0 gave it_1 to him_2 : $(\mathtt{Var} \to \top) \to \mathtt{t}$

(16) **variable-free**

she left : $\mathtt{e} \to \mathtt{t}$

she gave it to him : $\mathtt{e} \to \mathtt{e} \to \mathtt{e} \to \mathtt{t}$

(18) **polymorphic**

she_0 left : $(\mathtt{e}, \Gamma) \to \mathtt{t}$

she_0 gave it_1 to him_2 : $(\mathtt{e}, (\mathtt{e}, (\mathtt{e}, \Gamma))) \to \mathtt{t}$

The upshot of this is that — like traditional variable-free frameworks, and unlike traditional assignment-based frameworks — expressions with different kinds (and numbers of) dependencies have different types, signaling what anaphoric information is required for evaluation. But expressions are also uniform — like assignment-based frameworks, and unlike traditional variable-free frameworks — in their dependence on one (and only one) context, which may well be accessed by multiple distinct pronouns.

Formally, different anaphors may access different components of the context, so as is common, we distinguish them with indices. But, crucially, differently indexed pro-forms will have different types. A pronoun that reads from the first coordinate of a state requires only that the state has a first element (and returns whatever that element is); a pronoun that reads from the second coordinate requires that the state has at least two elements (and returns whatever the second element is); etc.

(19a) $\text{it}_0 : (\alpha, \Gamma) \to \alpha$

(19b) $\text{it}_1 : (\alpha, (\beta, \Gamma)) \to \beta$

(19c) $\text{it}_2 : (\alpha, (\beta, (\varsigma, \Gamma))) \to \varsigma$

The entire infinite family of pronouns could be lexically stipulated, but it may also be generated inductively if indices themselves are identified as functions from contexts to values (Danvy 1998, Fridlender & Indrika 2000). For instance, let $[\![\mathsf{z}]\!]$ be the function that projects the first element of a pair. And let $[\![\mathsf{s}]\!]$ be the function that, given a projection function n, returns the projection function that skips the first element of a pair and applies n to the remainder.

(20)
$$\mathsf{z} : (\alpha, \Gamma) \to \alpha \qquad \mathsf{s} : (\Gamma \to \tau) \to (\alpha, \Gamma) \to \tau$$
$$[\![\mathsf{z}]\!] \coloneqq \mathsf{fst} \qquad [\![\mathsf{s}]\!] \coloneqq \lambda n.\, n \circ \mathsf{snd}$$

Here it is easiest to imagine Γ as standing for a sequence of types $(\sigma_1, (\ldots, (\sigma_{n-1}, \sigma_n)))$, understood as the types of the things that have been stored for potential reference. Then the "indices" used for binding are just z and all its s-uccessors: $\{\mathsf{z}, \mathsf{s}\,\mathsf{z}, \mathsf{s}\,(\mathsf{s}\,\mathsf{z}), \mathsf{s}\,(\mathsf{s}\,(\mathsf{s}\,\mathsf{z})), \ldots\}$.

(21)
$$[\![\mathsf{z}]\!] = \mathsf{fst}$$
$$[\![\mathsf{s}\,\mathsf{z}]\!] = \mathsf{fst} \circ \mathsf{snd}$$
$$[\![\mathsf{s}\,(\mathsf{s}\,\mathsf{z})]\!] = \mathsf{fst} \circ \mathsf{snd} \circ \mathsf{snd}$$
$$[\![\mathsf{s}\,(\mathsf{s}\,(\mathsf{s}\,\mathsf{z}))]\!] = \mathsf{fst} \circ \mathsf{snd} \circ \mathsf{snd} \circ \mathsf{snd}$$
$$\ldots$$

The type of an index is determined by the types of z and s in (20), which together recurse through Γ until the index bottoms out at z. Whatever type is stored in that position is the type of the pronoun in that context. We thus replace an infinite class of lexical pronouns with a couple of grammatical constructs.

This would be sufficient for the sort of simple, static, variable-free fragment we will demonstrate in Section 3.2. But anticipating the transition to dynamic semantics in Section 5, we will need a little bit more from our indices. In particular, we will want to generalize the inductive hierarchy so that each index can be used either to access a particular coordinate of the state as in (21), or to modify that coordinate instead.

Dual read/write operators like this are sometimes referred to as *lenses* in programming language theory. For our purposes, we will take a lens to be a function of the form $\sigma \to (\alpha \times (\beta \to \tau))$. Any such function can be used as a projection from a state σ to a particular coordinate α, or alternatively as a means of converting an item α in a state σ to an item of type β, yielding a new state τ. These coercions are canonically called the *get* and *mod* operations of the lens. Formally, for any lens n, we have:

(22) $\mathsf{get}\, n \coloneqq \lambda s.\, \mathsf{fst}\,(n\, s)$

(23) $\mathsf{mod}\, n \coloneqq \lambda f.\lambda s.\, \mathsf{snd}\,(n\, s)\,(f\,(\mathsf{fst}\,(n\, s)))$

Different data structures σ call for different lenses. Since we are only interested in these nested tuples modeling records of discourse reference, we define all the lenses we could need inductively, just as above. The initial index 'Z' targets an entire context, metaphorically 'placing a read/write cursor' outside of the context. For any index n, the index '$\mathsf{R}\,n$' skips over the first element of a context, and uses n to target a coordinate in the remainder. Intuitively, this moves the cursor determined by n one step inward and to the right. Symmetrically, '$\mathsf{L}\,n$' moves the cursor determined by n one step to the left, though we will not have any need of this until Section 6.1.

(24)
$$\mathsf{Z} : \alpha \to (\alpha \times (\beta \to \beta))$$
$$[\![\mathsf{Z}]\!] \coloneqq \lambda a.\, \langle a, \lambda b.\, b\rangle$$

$$\mathsf{R} : (\Gamma \to (\alpha \times (\beta \to \mathtt{H}))) \to (\xi, \Gamma) \to (\alpha \times (\beta \to (\xi, \mathtt{H})))$$
$$[\![\mathsf{R}]\!] \coloneqq \lambda n.\lambda(x, g).\, \langle \mathsf{fst}\,(n\,g), \lambda b.\, (x, \mathsf{snd}\,(n\,g)\,b)\rangle$$

$$\mathsf{L} : (\Gamma \to (\alpha \times (\beta \to \mathtt{H}))) \to (\Gamma, \xi) \to (\alpha \times (\beta \to (\mathtt{H}, \xi)))$$
$$[\![\mathsf{L}]\!] \coloneqq \lambda n.\lambda(g, x).\, \langle \mathsf{fst}\,(n\,g), \lambda b.\, (\mathsf{snd}\,(n\,g)\,b, x)\rangle$$

Together, 'Z' and 'R' can be used to walk the spine of a heterogeneous list to target any finite position, as in (25).

(25)
$$[\![\mathsf{Z}]\!] = \lambda a.\, \langle a, \lambda b.\, b\rangle$$
$$[\![\mathsf{R}\,\mathsf{Z}]\!] = \lambda(x, g).\, \langle g, \lambda b.\, (x, b)\rangle$$
$$[\![\mathsf{R}\,(\mathsf{R}\,\mathsf{Z})]\!] = \lambda(x, (y, g)).\, \langle g, \lambda b.\, (x, (y, b))\rangle$$
$$[\![\mathsf{R}\,(\mathsf{R}\,(\mathsf{R}\,\mathsf{Z}))]\!] = \lambda(x, (y, (z, g))).\, \langle g, \lambda b.\, (x, (y, (z, b)))\rangle$$
$$\ldots$$

We may then recover the static, projective indices of the previous section from the getters of these lenses. The conversion is given by the function var in (26), which turns any lens index n built from R and Z into the corresponding projection function built from s and z, as shown in (27).

(26)
$$\mathsf{var}\,n = \lambda g.\, \mathsf{fst}\,(\mathsf{get}\,n\,g)$$

(27)
$$[\![\mathsf{z}]\!] \coloneqq \mathsf{var}\,[\![\mathsf{Z}]\!]$$
$$= \lambda(a, g).\, a$$
$$[\![\mathsf{s}\,\mathsf{z}]\!] \coloneqq \mathsf{var}\,[\![\mathsf{R}\,\mathsf{Z}]\!]$$
$$= \lambda(x, (a, g)).\, a$$
$$[\![\mathsf{s}\,(\mathsf{s}\,\mathsf{z})]\!] \coloneqq \mathsf{var}\,[\![\mathsf{R}\,(\mathsf{R}\,\mathsf{Z})]\!]$$
$$= \lambda(x, (y, (a, g))).\, a$$
$$[\![\mathsf{s}\,(\mathsf{s}\,(\mathsf{s}\,\mathsf{z}))]\!] \coloneqq \mathsf{var}\,[\![\mathsf{R}\,(\mathsf{R}\,(\mathsf{R}\,\mathsf{Z}))]\!]$$
$$= \lambda(x, (y, (z, (a, g)))).\, a$$
$$\ldots$$

Fittingly, it is only the capacity to update records that is new to the dynamic indices:

(28)
$$\mathsf{mod}\,[\![\mathsf{Z}]\!] = \lambda f.\lambda g.\, f\,g$$
$$\mathsf{mod}\,[\![\mathsf{R}\,\mathsf{Z}]\!] = \lambda f.\lambda(x, g).\, (x, f\,g)$$
$$\mathsf{mod}\,[\![\mathsf{R}\,(\mathsf{R}\,\mathsf{Z})]\!] = \lambda f.\lambda(x, (y, g)).\, (x, (y, f\,g))$$
$$\mathsf{mod}\,[\![\mathsf{R}\,(\mathsf{R}\,(\mathsf{R}\,\mathsf{Z}))]\!] = \lambda f.\lambda(x, (y, (z, g))).\, (x, (y, (z, f\,g)))$$
$$\ldots$$

3.2 A static, variable-free fragment

Let us define the syntax and semantics for a simple, familiar applicative grammar. An expression in this grammar is either a lexical item drawn from some set $\mathcal{L}$, a pronoun with an index, an abstraction $\triangleright E$ (discussed below), or a concatenation of two sub-expressions. An index is either z or the successor of another index. For readability, we abbreviate indices with numerals in the expected way, such that $\text{pro}_k \equiv \text{pro}\,(\underbrace{\text{s}\,(\ldots(\text{s}\ \text{z})\ldots)}_{k\text{-times}})$.

(29) $E = \mathcal{L} \mid \text{pro}\,N \mid \triangleright E \mid E\,E$
$N = \text{z} \mid \text{s}\,N,$ with $0 \equiv \text{z},\ 1 \equiv \text{sz},\ 2 \equiv \text{s}\,(\text{sz}), 3 \equiv \text{s}\,(\text{s}\,(\text{sz})), \ldots$

Semantically, all is as expected. Here, and in the rest of the paper, we give denotations as functions from derivations to semantic values. A derivation is a proof that a certain expression has a certain type. To each inference rule of the type logic, we attach a corresponding semantic rule, giving the denotation of the inference step as a function of the denotations of (the proofs of) its premises. For instance, a rule of the form in (30) should be interpreted as saying that if you have a proof that expression E_1 has type α and another proof that expression E_2 has type β, then you can conclude that the expression E_3 has type ς. (Note that [r] is just a name for the rule, for expository purposes.) And the denotation of this derivation that E_3 has type ς is given as a function of the denotations of the derivations that E_1 has type α and E_2 has type β. We will often abbreviate $[\![E_1 : \alpha]\!]$ and $[\![E_2 : \beta]\!]$ on the right-hand side of the equals sign as $[\![E_1]\!]$ and $[\![E_2]\!]$, with the understanding that we mean the denotations of these expressions at the types specified in the premises.

(30) $$\left[\!\!\left[\dfrac{E_1 : \alpha \quad E_2 : \beta}{E_3 : \varsigma}\,[\text{r}] \right]\!\!\right] \coloneqq \ldots [\![E_1 : \alpha]\!] \ldots [\![E_2 : \beta]\!] \ldots$$

In practice, such proofs are just upside-down type-annotated trees, and interpreting a proof as a function of its premises is just the same as interpreting a tree as a function of its daughters.

The denotations of lexical items are assumed to come from the lexicon. We write this as a nullary rule, and assume all the lexical items we'll introduce are typed as specified and defined in $\mathcal{L}$.

(31) $$\left[\!\!\left[\dfrac{}{E : \sigma}\,[\text{Lex}] \right]\!\!\right] \coloneqq \mathcal{L}\,(E : \sigma)$$

Following Jacobson (1999), we may distinguish the type of functional dependency introduced by a pronoun from that introduced by argument structure. So $\Gamma \Rightarrow \alpha$ is the type of an expression that will denote an α in a context of type Γ, while $\alpha \to \beta$ is the type of an expression that will denote a β if given an argument α. Semantically, both types corresponds to functions. Pronouns, therefore, denote identity functions; they take their denotations directly from the projection functions defined by $[\![\text{z}]\!]$ and $[\![\text{s}]\!]$ above.

(32) $\text{pro} : (\Gamma \to \alpha) \to \Gamma \Rightarrow \alpha$
$[\![\text{pro}]\!] \coloneqq \lambda n.\, n$

The interpretation of a concatenation takes one of two forms, depending on the types of the sub-expressions being concatenated. If they are context-independent ($\Rightarrow$-free) then one of them simply takes the other as an argument, as in (33). Alternatively, if the daughters demand contexts before returning functions and arguments, then they are combined via (34), which ensures that

both daughters are evaluated in the relevant context before one is applied to the other. Finally, an expression containing pronominal dependencies may be combined with an ordinary context-independent expression by coercing the latter into a constant function compatible with any context. This coercion is given as the unary rule in (35).

(33) $$\left[\!\!\left[\dfrac{F : \sigma \to \tau \quad X : \sigma}{FX : \tau}\,[/] \right]\!\!\right] \coloneqq [\![F]\!]\,[\![X]\!]$$
$$\left[\!\!\left[\dfrac{X : \sigma \quad F : \sigma \to \tau}{XF : \tau}\,[\backslash] \right]\!\!\right] \coloneqq [\![F]\!]\,[\![X]\!]$$

(34) $$\left[\!\!\left[\dfrac{F : \Gamma \Rightarrow \sigma \to \tau \quad X : \Gamma \Rightarrow \sigma}{FX : \Gamma \Rightarrow \tau}\,[\oslash] \right]\!\!\right] \coloneqq \lambda g.\,[\![F]\!]\,g\,([\![X]\!]\,g)$$
$$\left[\!\!\left[\dfrac{X : \Gamma \Rightarrow \sigma \quad F : \Gamma \Rightarrow \sigma \to \tau}{XF : \Gamma \Rightarrow \tau}\,[\otimes] \right]\!\!\right] \coloneqq \lambda g.\,[\![F]\!]\,g\,([\![X]\!]\,g)$$

(35) $$\left[\!\!\left[\dfrac{X : \sigma}{X : \Gamma \Rightarrow \tau}\,[\eta] \right]\!\!\right] \coloneqq \lambda g.\,[\![X]\!]$$

This is enough to derive the denotations of basic sentences with free pronouns. For instance, if the lexicon $\mathcal{L}$ includes at least $(\text{Mary} : \mathsf{e})$ and $(\text{called} : \mathsf{e} \to \mathsf{e} \to \mathsf{e} \to \mathsf{t})$ in its domain, then we can derive the denotations of the sentences in (18).

(36a)
$$\dfrac{\dfrac{\dfrac{}{\text{called} : \mathsf{e} \to \mathsf{e} \to \mathsf{t}}\,[\text{Lex}]}{\text{called} : (\mathsf{e}, \Gamma) \Rightarrow \mathsf{e} \to \mathsf{e} \to \mathsf{t}}\,[\eta] \quad \dfrac{}{\text{pro}_0 : (\mathsf{e}, \Gamma) \Rightarrow \mathsf{e}}\,[z]}{\text{called}\,\text{pro}_0 : (\mathsf{e}, \Gamma) \Rightarrow \mathsf{e} \to \mathsf{t}}\,[\oslash]$$

(36b)
$$\dfrac{\dfrac{\dfrac{}{\text{Mary} : \mathsf{e}}\,[\text{Lex}]}{\text{Mary} : (\mathsf{e}, \Gamma) \Rightarrow \mathsf{e}}\,[\eta] \quad \begin{array}{c}\vdots \\ \text{called}\,\text{pro}_0 : (\mathsf{e}, \Gamma) \Rightarrow \mathsf{e} \to \mathsf{t}\end{array}}{\text{Mary}\,\text{called}\,\text{pro}_0 : (\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}}\,[\otimes]$$

(36c) $$[\![(36b)]\!] = \lambda g.\,\mathbf{call}\,(\mathrm{fst}\,g)\,\mathbf{m}$$

The derivation in (36b) demonstrates the effect of a free pronoun on the type of an expression. What remains is to show that this type-theoretic signature disappears once the pronoun is bound. To this end, let us introduce the type and semantics for a simple abstraction operator $\triangleright$ in (37). It is worth noting the type-theoretic push and pull between s and $\triangleright$. The former takes an expression N that depends on a context Γ, and yields an expression $\mathsf{s}\,N$ with a new, additional dependency (α, Γ); the latter takes an expression M which already depends on some object α of the context (α, Γ), and converts that contextual dependency into an ordinary functional argument, thereby loosening the requirements on the context Γ. The semantics of $\triangleright$ should of course look quite similar to that of any object-language abstraction operator. In fact, the definitions in (20), (34) and (37) are exactly Carette, Kiselyov & Shan's (2009) and Kiselyov's (2012) semantics for the simply-typed lambda calculus with de Bruijn indices in lieu of variables.

(37) $$\left[\!\!\left[\dfrac{M : (\alpha, \Gamma) \Rightarrow \tau}{\triangleright M : \Gamma \Rightarrow \alpha \to \tau}\,[\triangleright] \right]\!\!\right] \coloneqq \lambda g.\,\lambda a.\,[\![M]\!]\,(a, g)$$

For instance, (36b) may be continued as in (38a) to derive the sentence 'John, Mary called', construing pro_0 as the trace of the topicalized object 'John'.

(36b)

(38a)
$$\dfrac{\dfrac{\dfrac{}{\text{John} : \mathsf{e}}\,[\text{Lex}]}{\text{John} : \Gamma \Rightarrow \mathsf{e}}\,[\eta] \quad \dfrac{\begin{array}{c}\vdots\\ \text{Mary called pro}_0 : (\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}\end{array}}{\triangleright\ \text{Mary called pro}_0 : \Gamma \Rightarrow \mathsf{e} \to \mathsf{t}}\,[\triangleright]}{\text{John} \triangleright \text{Mary called pro}_0 : \Gamma \Rightarrow \mathsf{t}}\,[\oslash]$$

(38b) $[\![(38a)]\!] = \lambda g.\,\mathbf{call\ j\ m}$

Being a variant of the lambda calculus, it is plain that the fragment defined in (34) and (37) guarantees that there is no binding without c-command. All anaphors find their referents in the context, and the only opportunity for an expression to manipulate the context is through $\triangleright$. Likewise, even more plainly, an expression may only quantify over an expression it is adjacent to, since the only mode of combination is Function Application. We note that this is naturally exactly analogous to the way binding works in a typical static variable-full grammar like the Logical Forms of Heim & Kratzer (1998), which is also a variant of the lambda calculus, as well as the variable-free grammars of Jacobson (1999, 2014, et seq.).

Scaling this up to a full grammatical framework for anaphora requires relaxation along both of these dimensions. There must be a way for expressions to take as arguments other expressions that contain them, and a way for antecedents to bind pro-forms across phrases. For example, in (39) and (40), the properties that everyone is claimed to have are those of being talked to by Mary and having a mother who talked to Mary, respectively. Both of these are assembled from constituents dominating the universal. And the sentences in (41) and (42) have readings in which the referent of 'them' co-varies with the indefinite in the preceding clause.

(39) Mary talked to everyone

(40) Everyone's mother talked to Mary

(41) John saw a student and told them to talk to Mary

(42) Everyone who saw a student told them to talk to Mary

In other words, we need to enable at least some amount of scope and binding without c-command. But of course, there be a few dragons here. Perhaps most famously, we have the *crossover* paradigm in (43)–(46).

(43) Everyone's mother told someone to talk to Mary

(44) Everyone's mother told them to talk to Mary

(45) Someone's mother told everyone to talk to Mary

(46) *Their mother told everyone to talk to Mary

In (43), the quantifier embedded in the subject phrase can take scope over the verb phrase that it does not c-command, and accordingly it can bind pronouns in that verb phrase, as in (44). Likewise the quantifier embedded in the verb phrase of (45) can take scope over the subject that it does not c-command, but this time it *cannot* bind a pronoun in that subject (46). Consequently, we take it as a minimum requirement of a theory of composition that scope-taking feed binding, so long as the expression taking scope is to the left of the anaphor it is to bind (Barker 2012).

4 Continuations

Our starting point for integrating a theory of scope with the variable-free anaphoric semantics above is Shan & Barker's (2006) continuation-passing mode of combination. The technique provides an attractive theory of inverse scope, crossover, and reconstruction, and the authors offer compelling criticisms of the ways in which other static and dynamic theories have built in these sorts of evaluation order asymmetries. The reader is referred to Barker & Shan 2008 and Barker & Shan 2014 for summaries, applications, and tutorial introductions to the mechanism. For our purposes, we need only introduce the following two technical notions.

First, essentially by definition, expressions that take scope have higher-order denotations, with types of the form $(\alpha \to o) \to \rho$. That initial argument, called the *continuation*, or scope, of the expression, intuitively encodes the function mapping lower-typed objects $a : \alpha$ to whatever the denotation of the enclosing syntactic environment o would be if the expression were replaced by a. We reserve a distinguished type constructor for denotations of this form.

(47) $$\frac{\rho \mid o}{\alpha} \coloneqq (\alpha \to o) \to \rho$$

For instance, ordinary Generalized Quantifiers have type $\frac{t \mid t}{e}$. They act locally like entities e, they take scope over expressions that are either true or false of an entity, and they use this continuation, of type $e \to t$, to compute a truth value t. For an expression of type $\frac{\rho \mid o}{\alpha}$, we will sometimes refer to the type α as the expression's *local*, or *underlying*, type. Again, loosely speaking, this is the type that one would expect to find in the position where the scope-taking expression occurs.

Second, two expressions that both expect continuations may be composed to form a new continuation-expecting expression, so long as their underlying types can be composed. Formally, the rule schemas in (48) and (49) generalize any unary and binary modes of combination to higher-order functions of the appropriate types.

(48) $$\text{If } \left[\!\!\left[\frac{T : \sigma}{E : \upsilon}[\dagger] \right]\!\!\right] = \dagger\,[\![C]\!], \text{ then } \left[\!\!\left[\dfrac{T : \frac{\rho \mid o}{\sigma}}{E : \frac{\rho \mid o}{\upsilon}}[\text{K-}\dagger] \right]\!\!\right] = \lambda k.\,[\![T]\!]\,(\lambda x.\,k\,(\dagger\, x)),$$

(49) $$\text{If } \left[\!\!\left[\frac{L : \sigma \quad R : \tau}{E : \upsilon}[*] \right]\!\!\right] = [\![L]\!] * [\![R]\!],$$
$$\text{then } \left[\!\!\left[\dfrac{L : \frac{\rho \mid \pi}{\sigma} \quad R : \frac{\pi \mid o}{\tau}}{E : \frac{\rho \mid o}{\upsilon}}[\text{K-}*] \right]\!\!\right] = \lambda k.\,[\![L]\!]\,(\lambda l.\,[\![R]\!]\,(\lambda r.\,k\,(l * r))),$$

For instance, the unary rule in (35) gives rise to the continuation-passing rule in (50). And the binary rules in (33) and (34) give rise to the rules in (51) and (52).

(50) $$\left[\!\!\left[\dfrac{X : \dfrac{\rho \mid o}{\sigma}}{X : \dfrac{\rho \mid o}{\Gamma \Rightarrow \sigma}}\,[\text{K-}\eta] \right]\!\!\right] \coloneqq \lambda k.\, [\![X]\!]\,(\lambda x.\, k\,(\lambda g.\, x))$$

(51) $$\left[\!\!\left[\dfrac{F : \dfrac{\rho \mid \pi}{\sigma \to \tau} \quad X : \dfrac{\pi \mid o}{\sigma}}{FX : \dfrac{\rho \mid o}{\tau}}\,[\text{K-}/] \right]\!\!\right] \coloneqq \lambda k.\, [\![F]\!]\,(\lambda f.\, [\![X]\!]\,(\lambda x.\, k\,(f\,x)))$$

(52) $$\left[\!\!\left[\dfrac{F : \dfrac{\rho \mid \pi}{\Gamma \Rightarrow \sigma \to \tau} \quad X : \dfrac{\pi \mid o}{\Gamma \Rightarrow \sigma}}{FX : \dfrac{\rho \mid o}{\Gamma \Rightarrow \tau}}\,[\text{K-}\varnothing] \right]\!\!\right] \coloneqq \lambda k.\, [\![F]\!]\,(\lambda f.\, [\![X]\!]\,(\lambda x.\, k\,(\lambda g.\, f\,g\,(x\,g))))$$

Note that because the schemas in (48) and (49) generate new unary and binary modes of combination, they feed themselves. Thus we have a hierarchy of derived rules for combining increasingly highly-typed expressions. For instance, applying (48) to (50) determines the rule in (53). Likewise for the doubly-lifted version of function application in (54). And so on.

(53) $$\left[\!\!\left[\dfrac{T : \dfrac{\dfrac{\rho' \mid o'}{\rho \mid o}}{\sigma}}{T : \dfrac{\dfrac{\rho' \mid o'}{\rho \mid o}}{\Gamma \Rightarrow \sigma}}\,[\text{K}^2\text{-}\eta] \right]\!\!\right] \coloneqq \lambda c.\, [\![T]\!]\,(\lambda T'.\, c\,(\lambda k.\, T'\,(\lambda x.\, k\,(\lambda g.\, x))))$$

(54) $$\left[\!\!\left[\dfrac{L : \dfrac{\dfrac{\rho' \mid \pi'}{\rho \mid \pi}}{\sigma \to \tau} \quad R : \dfrac{\dfrac{\pi' \mid o'}{\pi \mid o}}{\sigma}}{LR : \dfrac{\dfrac{\rho' \mid o'}{\rho \mid o}}{\tau}}\,[\text{K}^2\text{-}/] \right]\!\!\right] \coloneqq \lambda c.\, [\![L]\!]\,(\lambda L'.\, [\![R]\!]\,(\lambda R'.\, c\,(\lambda k.\, L'\,(\lambda f.\, R'\,(\lambda x.\, k\,(f\,x))))))$$

Finally, we must say how an ordinary expression of type σ may be combined with continuation-taking expressions whose underlying type would fit together with σ. For this, Shan & Barker introduce the operators **U** and **D**, which take expressions in and out of the scope-taking mode.

(55) $$\left[\!\!\left[\dfrac{E : \sigma}{E : \dfrac{\rho \mid \rho}{\sigma}}\,[\mathbf{U}] \right]\!\!\right] \coloneqq \lambda k.\, k\,[\![E]\!]$$

(56) $$\left[\!\!\left[\dfrac{E : \dfrac{\rho \mid \tau}{\tau}}{E : \rho}\,[\mathbf{D}] \right]\!\!\right] \coloneqq [\![E]\!]\,(\lambda t.\, t)$$

These operators, together with the continuation-passing rules induced by (48)/(49), and the basic pronoun (20), application (34), and abstraction (37) rules above, suffice to generate a wealth of scope and binding configurations.

(57) $$\left[\!\!\left[\text{everyone} : \dfrac{\Gamma \Rightarrow \mathsf{t} \mid (\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}}{\mathsf{e}} \right]\!\!\right] \coloneqq \lambda k.\, \lambda g.\, \forall x.\, k\, x\, (x, g)$$

(58a)
$$\dfrac{\dfrac{\dfrac{}{\text{everyone's} : \frac{\Gamma \Rightarrow \mathsf{t} \mid (\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}}{\mathsf{e}}}[\text{Lex}] \quad \dfrac{\dfrac{}{\text{mom} : \mathsf{e} \to \mathsf{e}}[\text{Lex}]}{\text{mom} : \frac{\rho \mid \rho}{\mathsf{e} \to \mathsf{e}}}[\mathbf{U}]}{\text{eo's mom} : \frac{\Gamma \Rightarrow \mathsf{t} \mid (\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}}{\mathsf{e}}}[\text{K-}\backslash] \quad \text{(36a)}}{\text{eo's mom} : \frac{\Gamma \Rightarrow \mathsf{t} \mid (\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}}{(\mathsf{e}, \Gamma) \Rightarrow \mathsf{e}}}[\text{K-}\eta] \quad \dfrac{\text{called pro}_0 : (\mathsf{e}, \Gamma) \Rightarrow \mathsf{e} \to \mathsf{t}}{\text{called pro}_0 : \frac{\rho \mid \rho}{(\mathsf{e}, \Gamma) \Rightarrow \mathsf{e} \to \mathsf{t}}}[\mathbf{U}]}{\text{eo's mom called pro}_0 : \frac{\Gamma \Rightarrow \mathsf{t} \mid (\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}}{(\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}}}[\text{K-}\oslash]}{\text{eo's mom called pro}_0 : \Gamma \Rightarrow \mathsf{t}}[\mathbf{D}]$$

(58b) $$[\![(58a)]\!] = \lambda g.\, \forall x.\, \mathbf{call}\, x\, (\mathbf{mom}\, x)$$

Notice that the possessor here, 'everyone', binds the pronoun 'pro_0' despite not c-commanding it. This is possible because the quantifier adds its referent to the environment that its scope is evaluated in, and the pronoun falls into that scope. The relative derivational dominance of the two items is made moot.

Unfortunately, as things stand, so is their linear order. The derivation in (59b) comes out equivalent to that in (58a).

(59a)
$$\dfrac{\dfrac{\dfrac{\dfrac{}{\text{called} : \mathsf{e} \to \mathsf{e} \to \mathsf{t}}[\text{Lex}]}{\text{called} : \frac{\rho \mid \rho}{\mathsf{e} \to \mathsf{e} \to \mathsf{t}}}[\mathbf{U}] \quad \dfrac{}{\text{everyone} : \frac{\Gamma \Rightarrow \mathsf{t} \mid (\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}}{\mathsf{e}}}[\text{Lex}]}{\text{called eo} : \frac{\Gamma \Rightarrow \mathsf{t} \mid (\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}}{\mathsf{e} \to \mathsf{t}}}[\text{K-}/]}{\text{called eo} : \frac{\Gamma \Rightarrow \mathsf{t} \mid (\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}}{(\mathsf{e}, \Gamma) \Rightarrow \mathsf{e} \to \mathsf{t}}}[\text{K-}\eta]$$

(59a)

$$\dfrac{\dfrac{\dfrac{}{\text{pro}_0\text{'s} : (\mathsf{e}, \Gamma) \Rightarrow \mathsf{e}}[z] \quad \dfrac{\dfrac{}{\text{mom} : \mathsf{e} \to \mathsf{e}}[\text{Lex}]}{\text{mom} : (\mathsf{e}, \Gamma) \Rightarrow \mathsf{e} \to \mathsf{e}}[\eta]}{\text{pro}_0\text{'s mom} : (\mathsf{e}, \Gamma) \Rightarrow \mathsf{e}}[\oslash]}{\text{pro}_0\text{'s mom} : \dfrac{\rho \mid \rho}{(\mathsf{e}, \Gamma) \Rightarrow \mathsf{e}}}[\mathbf{U}] \qquad \begin{array}{c}\\ \vdots \\ \text{called eo} : \dfrac{\Gamma \Rightarrow \mathsf{t} \mid (\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}}{(\mathsf{e}, \Gamma) \Rightarrow \mathsf{e} \to \mathsf{t}}\end{array}}{\text{pro}_0\text{'s mom called eo} : \dfrac{\Gamma \Rightarrow \mathsf{t} \mid (\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}}{(\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}}}[\text{K-}\oslash]}{\text{pro}_0\text{'s mom called eo} : \Gamma \Rightarrow \mathsf{t}}[\mathbf{D}]$$

(59b)

(59c) $[\![(59\text{b})]\!] = \lambda g.\, \forall x.\, \mathbf{call}\, x\, (\mathbf{mom}\, x)$

Shan & Barker's (2006) solution to this problem is twofold. First, they use the transparency provided by the type system to restrict the use of **D** to expressions whose underlying types are closed. Intuitively, this means expressions cannot scope over constituents with free pronouns in them. Formally, they give **D** the much stricter type in (60), which operates only on constituents with the concrete underlying type `t`. The restriction to underlying type `t` guarantees that only context-independent clauses may serve as the continuation to an expression that has taken scope.

(60) $$\left[\!\!\left[\dfrac{E : \dfrac{\rho \mid \mathsf{t}}{\mathsf{t}}}{E : \rho}[\mathbf{D}] \right]\!\!\right] \coloneqq [\![E]\!]\,(\lambda a.\, a)$$

In the fragment we have outlined so far, this typing of **D** is slightly stricter than is required. For one, there is nothing particularly special about the type `t`, as far as preventing crossover goes. Really all that is required is that it not harbor free pronouns. To that end, any concrete, $\Rightarrow$-free type would be fine. Second, it does not matter whether the underlying type is *formally* context-sensitive (begins with $\_\_ \Rightarrow$) or not; it just needs to be the case that the underlying type does not *depend* on the shape of the context in any way. Any particular pronoun will place a particular constraint on what its context should look like. Only a closed expression — one with no free pronouns — can afford not to care. So any constituent ought to able to serve as the scope of an expression so long as it places no demands on its anaphoric context, or equivalently, as long as it is compatible with any anaphoric context whatsoever, including the empty context $()$. So with an eye toward the dynamic analysis in Section 5, we offer the variant of Shan & Barker's lowering operation in (61).

(61) $$\left[\!\!\left[\dfrac{E : \dfrac{\rho \mid \Gamma \Rightarrow \tau}{() \Rightarrow \tau}}{E : \rho}[\mathbf{D}] \right]\!\!\right] \coloneqq [\![E]\!]\,(\lambda m.\lambda g.\, m\,())$$

This version of **D** empties the local context before evaluating its scope, passing in $()$ instead. This effectively thwarts the efforts of the universal to pass in the augmented context (x, g) that might value a free pronoun. Indeed, the only sorts of expressions evaluable in such amnesiac contexts are those with no free pronouns at all.

With **D** so restricted, the derivation in (59b) is no longer valid. The final lowering step does not have a type consistent with the **D** rule, since its underlying type is context dependent: $(\mathsf{e}, ()) \Rightarrow \mathsf{t}$. Sadly this also invalidates the derivation in (58a), since it has exactly the same final step.

So the second half of Shan & Barker's (2006) solution is to lift the type of pronouns so that they too become scope-takers. They do this in the lexical semantics of the pronoun, but again anticipating Section 5, we offer a more modular approach, supplementing our compositional regime with [★]. Together, η and $\star$ comprise a *monad* (Charlow 2014, 2019b).

(62) $$\left[\!\!\left[\dfrac{E : \Gamma \Rightarrow \sigma}{E : \dfrac{\Gamma \Rightarrow \tau \,\middle|\, \Gamma \Rightarrow \tau}{\sigma}}[\star] \right]\!\!\right] \coloneqq \lambda k.\lambda g.\, k\,(\llbracket E \rrbracket\, g)\, g$$

The [★] rule converts an expression of type $\Gamma \Rightarrow \sigma$ to an expression of type $\dfrac{\Gamma \Rightarrow \tau \,\middle|\, \Gamma \Rightarrow \tau}{\sigma}$. It is to context-dependent expressions what **U** is to pure expressions. This can be seen by simply ignoring all of the $\Gamma \Rightarrow$'s in the types and g's in the denotation.[1] This opens up a second derivation of the verb phrase in (58a), shown in (63a). The key is the last step of (63b). Just before this step, the underlying type of the sentence is completely unconstrained in its choice of environment Γ'. Since Γ' is just a type variable, it is free to unify with $()$ so that the [**D**] rule can apply.

(63a)
$$\dfrac{\dfrac{\dfrac{}{\text{called} : \mathsf{e} \to \mathsf{e} \to \mathsf{t}}[\text{Lex}]}{\text{called} : \dfrac{\rho \,\middle|\, \rho}{\mathsf{e} \to \mathsf{e} \to \mathsf{t}}}[\mathbf{U}] \qquad \dfrac{\dfrac{}{\text{pro}_0 : (\mathsf{e}, \Gamma) \Rightarrow \mathsf{e}}[z]}{\text{pro}_0 : \dfrac{(\mathsf{e}, \Gamma) \Rightarrow \mathsf{t} \,\middle|\, (\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}}{\mathsf{e}}}[\star]}{\text{called pro}_0 : \dfrac{(\mathsf{e}, \Gamma) \Rightarrow \mathsf{t} \,\middle|\, (\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}}{\mathsf{e} \to \mathsf{t}}}[\text{K-}\backslash]$$

(63b)
$$\dfrac{\dfrac{\dfrac{\dfrac{}{\text{eo's} : \dfrac{\Gamma \Rightarrow \mathsf{t} \,\middle|\, (\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}}{\mathsf{e}}}[\text{Lex}] \qquad \dfrac{\dfrac{}{\text{mom} : \mathsf{e} \to \mathsf{e}}[\text{Lex}]}{\text{mom} : \dfrac{\rho \,\middle|\, \rho}{\mathsf{e} \to \mathsf{e}}}[\mathbf{U}]}{\text{eo's mom} : \dfrac{\Gamma \Rightarrow \mathsf{t} \,\middle|\, (\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}}{\mathsf{e}}}[\text{K-}\backslash] \qquad \begin{array}{c}(63a)\\ \vdots \\ \text{called pro}_0 : \dfrac{(\mathsf{e}, \Gamma) \Rightarrow \mathsf{t} \,\middle|\, (\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}}{\mathsf{e} \to \mathsf{t}}\end{array}}{\text{eo's mom called pro}_0 : \dfrac{\Gamma \Rightarrow \mathsf{t} \,\middle|\, (\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}}{\mathsf{t}}}[\text{K-}\backslash]}[\text{K-}\eta]}{\text{eo's mom called pro}_0 : \dfrac{\Gamma \Rightarrow \mathsf{t} \,\middle|\, (\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}}{\Gamma' \Rightarrow \mathsf{t}}}[\mathbf{D}]$$

(63c) $\llbracket (63b) \rrbracket = \lambda g.\, \forall x.\, \mathbf{call}\, x\, (\mathbf{mom}\, x)$

Of course, a hopped-up derivation analogous to (59b) is also possible. But it does not result in any binding. Because the pronoun occurs to the left of the quantifier that would bind it, it simply

1 Note that the binary rules in (34) become otiose in the presence of [★], [η], and [K-/]. Any derivational step $\dfrac{F \quad X}{FX}[\oslash]$ can be replaced by $\dfrac{\dfrac{\dfrac{F}{F}[\star] \quad \dfrac{X}{X}[\star]}{FX}[\text{K-}/]}{\dfrac{FX}{FX}[\text{K-}\eta]}[\mathbf{D}]$ with no change in type or meaning.

outscopes that quantifier. As a result, the modification that 'everyone' makes to its local context does not amount to anything, and the anaphoric dependence introduced by the pronoun lives on. This is shown in (64b). Alternatively, if the object quantifier were lifted to take inverse scope over the pronoun, the derivation would become un-lowerable, just like (59b).

(64a)
$$\dfrac{\dfrac{\dfrac{}{\text{called} : \mathsf{e} \to \mathsf{e} \to \mathsf{t}}\,[\text{Lex}]}{\text{called} : \dfrac{\rho \,\big|\, \rho}{\mathsf{e} \to \mathsf{e} \to \mathsf{t}}}\,[\mathbf{U}] \qquad \dfrac{}{\text{everyone} : \dfrac{\Gamma \Rightarrow \mathsf{t} \,\big|\, (\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}}{\mathsf{e}}}\,[\text{Lex}]}{\text{called eo} : \dfrac{\Gamma \Rightarrow \mathsf{t} \,\big|\, (\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}}{\mathsf{e} \to \mathsf{t}}}\,[\text{K-/}]$$

(64b)
$$\dfrac{\dfrac{\dfrac{\dfrac{}{\text{pro}_0\text{'s} : (\mathsf{e}, \Gamma) \Rightarrow \mathsf{e}}\,[z]}{\text{pro}_0 : \dfrac{(\mathsf{e}, \Gamma) \Rightarrow \mathsf{t} \,\big|\, (\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}}{\mathsf{e}}}\,[\star] \qquad \dfrac{\dfrac{}{\text{mom} : \mathsf{e} \to \mathsf{e}}\,[\text{Lex}]}{\text{mom} : \dfrac{\rho \,\big|\, \rho}{\mathsf{e} \to \mathsf{e}}}\,[\mathbf{U}]}{\text{pro}_0\text{'s mom} : \dfrac{(\mathsf{e}, \Gamma) \Rightarrow \mathsf{t} \,\big|\, (\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}}{\mathsf{e}}}\,[\text{K-}\backslash] \qquad \begin{array}{c}(64a)\\ \vdots \\ \text{called eo} : \dfrac{\Gamma \Rightarrow \mathsf{t} \,\big|\, (\mathsf{e}, \Gamma) \Rightarrow \mathsf{t}}{\mathsf{e} \to \mathsf{t}}\end{array}}{\dfrac{\dfrac{\dfrac{\text{pro}_0\text{'s mom called eo} : \dfrac{(\mathsf{e}, \Gamma) \,\big|\, (\mathsf{e}, (\mathsf{e}, \Gamma)) \Rightarrow \mathsf{t}}{\mathsf{t}}}{\text{pro}_0\text{'s mom called eo} : \dfrac{(\mathsf{e}, \Gamma) \,\big|\, (\mathsf{e}, (\mathsf{e}, \Gamma)) \Rightarrow \mathsf{t}}{\Gamma' \Rightarrow \mathsf{t}}}\,[\text{K-}[\eta]]}{\text{pro}_0\text{'s mom called eo} : (\mathsf{e}, \Gamma)}\,[\mathbf{D}]}\,[\text{K-}\backslash]$$

(64c) $[\![(64b)]\!] = \lambda(a, g).\, \forall x.\, \mathbf{call}\, x\, (\mathbf{mom}\, a)$

This account of crossover has several attractive qualities. It is variable-free, so not tied to any particular indexing scheme or representational assumptions. It explains crossover in terms of general left-to-right evaluation order, but unlike dynamic accounts that purport to do the same, it includes a full-fledged treatment of scope inversion. And it lends itself to elegant treatments of reconstruction which would otherwise appear to counterexemplify the crossover pattern in (43)–(46).

But it is fundamentally a *static* semantics, in that it yokes binding to scope. That is, in order for a binder to value a pronoun, that pronoun must be part of the continuation passed in as an argument to the binder. If the binder is quantificational, then the pronoun must occur as part of the content that the binder semantically quantifies over. These assumptions face *prima facie* challenges from standard cross-clausal and donkey configurations.

(65) John saw a student and told them to talk to Mary

(66) Everyone who saw a student told them to talk to Mary

At the same time, even as pronouns may co-vary with indefinites that do not scope over them, they still generally resist binding from anything to their right. Thus we have the "secondary" crossover paradigm in (67).

(67a) Some farmer described every donkey they own to Mary

(67b) Mary described some donkey to every farmer who asked about it

(67c) Every farmer who owns a donkey described it to Mary

(67d) *Mary described it to every farmer who owns a donkey

Barker & Shan 2008 have pursued an inventive static account of (65)–(67), maintaining the connection between scope and binding, and eschewing traditional dynamic explanations for the relationship between indefinites and downstream pronouns. Here we ask if it is possible to go the other way, to relax the connection between scope and binding in the familiar spirit of dynamic semantics, while maintaining the idea that crossover is ultimately a consequence of the way that scope-takers combine with their scopes. We do this partly because of theoretical and empirical objections raised against the Barker & Shan 2008 account of donkey anaphora, and partly to establish that in-scope binding is not explanatorily essential to a genuinely semantic (variable-free) analysis of crossover.

5 Indexed state

Leveling up from a static to a dynamic semantics is, at a minimum, as simple as replacing denotations that read from a context with denotations that both read from and return a context. That is, where in the previous section we might have had an expression of type $\iota \to \alpha$, we should now expect to find an expression of type $\iota \to \alpha \times o$. But given the special role that indefinites play in discourse, nearly all dynamic frameworks for anaphora treat the transition from input states to output states as relational rather than functional. Thus we actually want to replace expressions of type $\iota \to \alpha$ with expressions of type $\iota \to \{\alpha \times o\}$. We notate such nondeterministic transitions as $\iota \overset{\alpha}{\Rightarrow} o$.

Unsurprisingly, the paradigmatic denotations exemplifying this dynamic type are those of indefinites. Let us assume (for a second) that DPs like 'a cat' always update the discourse by adding an entity to the context for downstream anaphora. Of course which entity is added is a matter of uncertainty, so the updates happen in parallel so to speak, one per potential witness to the indefinite description. This leads to denotations like (68). The type $\Gamma \overset{e}{\Rightarrow} (e, \Gamma)$ reveals the context updating nature of the expression; if the context has type Γ before the expression is evaluated, it will have type (e, Γ) after it is evaluated, no matter what type Γ is.

(68) $\text{a cat} : \Gamma \overset{e}{\Rightarrow} (e, \Gamma)$
$[\![\text{a cat}]\!] \coloneqq \lambda g.\, \{\langle x, (x, g)\rangle \mid \mathbf{cat}\, x\}$

(69) $\text{every cat} : \dfrac{\Gamma \overset{t}{\Rightarrow} \Gamma \,\Big|\, (e, \Gamma) \overset{t}{\Rightarrow} H}{e}$
$[\![\text{every cat}]\!] \coloneqq \lambda k \lambda g.\, \{\langle \forall x.\, \mathbf{cat}\, x \to \exists h.\, \langle \mathbf{T}, h\rangle \in k\, x\, (x, g), g\rangle\}$

The architecture of the grammar presented in Sections 3 and 4 needn't change at all. We simply upgrade any rule involving context-dependent denotations so that it may record and/or pass on contexts (potentially nondeterministically) in addition to reading from them. For the η and $\star$ rules, this is mostly just a matter of type plumbing.

(70) $$\left[\!\!\left[\dfrac{E : \tau}{E : \Gamma \overset{\tau}{\Rightarrow} \Gamma}[\eta] \right]\!\!\right] \coloneqq \lambda g.\, \{\langle [\![E]\!], g\rangle\}$$

(71) $$\left[\!\!\left[\dfrac{E : \Gamma \overset{\sigma}{\Rightarrow} H}{E : \dfrac{\Gamma \overset{\tau}{\Rightarrow} \Theta \,\Big|\, H \overset{\tau}{\Rightarrow} \Theta}{\sigma}}[\star] \right]\!\!\right] \coloneqq \lambda k.\lambda g.\, \bigcup\{k\, x\, h \mid \langle x, h\rangle \in [\![E]\!]\, g\}$$

While the static η and $\star$ operations form a monad (as mentioned in Section 4), the dynamic versions of these operations can be used to build and sequence computations parameterized to input and output types, which can crucially differ (note the Γ, $\mathtt{H}$, and Θ in the entry for $\star$). This is a consequence of our contention that dref introduction and consumption both result in changes to the type of the discourse record. As such, the dynamic η and $\star$ instantiate a more general construction than a monad, variously known as a *parameterized* or *indexed* monad (Wadler 1994, Atkey 2009).

The abstraction rule [▷] in (37) is engineered to convert a constituent containing a pronoun into a nuclear scope, binding that pronoun by adding the scope's argument to the local context in which its prejacent is evaluated. As discussed above, this has the consequence that binding is only possible from an argument to its scope ("in-scope binding"), since it is the argument of the abstraction $\triangleright E$ that provides the relevant referent for the pronoun in E. In upgrading this rule, we take advantage of the new dynamics. The rule in (72) simply augments a context Γ with the denotation of an arbitrary expression E. Since this outgoing state containing the referent of E will persist, there is no need for E to interact directly with a constituent containing a to-be-bound pronoun.

$$(72)\qquad \left[\!\!\left[\frac{E : \alpha}{E^{\triangleright} : \Gamma \overset{\alpha}{\Rightarrow} (\alpha, \Gamma)}\,[\triangleright] \right]\!\!\right] \coloneqq \lambda g.\,\{\langle [\![E]\!], ([\![E]\!], g)\rangle\}$$

Nothing about the function application or continuation-passing rules for scope-taking were specific to context-dependence, and they can be reused without any ado here. The same is true of the lifting type-shifter **U**. All that remains is to discuss the semantics of pronouns, and then to adjust the lowering operator **D** along the lines of (70)–(72) above.

5.1 Dynamic pronouns

Recall the polymorphic index functions in Section 3.1, repeated here in (73)–(74).

$$(73)\qquad \begin{array}{l} \mathsf{Z} : \alpha \to (\alpha \times (\beta \to \beta)) \\ [\![\mathsf{Z}]\!] \coloneqq \lambda a.\,\langle a, \lambda b.\, b\rangle \end{array}$$

$$\begin{array}{l} \mathsf{R} : (\Gamma \to (\alpha \times (\beta \to \mathtt{H}))) \to (\xi, \Gamma) \to (\alpha \times (\beta \to (\xi, \mathtt{H}))) \\ [\![\mathsf{R}]\!] \coloneqq \lambda n.\lambda(x, g).\,\langle \mathsf{fst}\,(n\,g), \lambda b.\,(x, \mathsf{snd}\,(n\,g)\,b)\rangle \end{array}$$

$$\begin{array}{l} \mathsf{L} : (\Gamma \to (\alpha \times (\beta \to \mathtt{H}))) \to (\Gamma, \xi) \to (\alpha \times (\beta \to (\mathtt{H}, \xi))) \\ [\![\mathsf{L}]\!] \coloneqq \lambda n.\lambda(g, x).\,\langle \mathsf{fst}\,(n\,g), \lambda b.\,(\mathsf{snd}\,(n\,g)\,b, x)\rangle \end{array}$$

$$(74)\qquad \begin{array}{l} [\![\mathsf{Z}]\!] = \lambda a.\,\langle a, \lambda b.\, b\rangle \\ [\![\mathsf{R}\,\mathsf{Z}]\!] = \lambda(x, g).\,\langle g, \lambda b.\,(x, b)\rangle \\ [\![\mathsf{R}\,(\mathsf{R}\,\mathsf{Z})]\!] = \lambda(x, (y, g)).\,\langle g, \lambda b.\,(x, (y, b))\rangle \\ [\![\mathsf{R}\,(\mathsf{R}\,(\mathsf{R}\,\mathsf{Z}))]\!] = \lambda(x, (y, (z, g))).\,\langle g, \lambda b.\,(x, (y, (z, b)))\rangle \\ \ldots \end{array}$$

Up to this point, we have only made use of the get functionality of these lenses, treating them as an inductively defined family of projection functions. But the dynamic lexical entry for pronouns, given in (75), finally takes advantage of their state-updating functionality as well.

$$(75)\qquad \begin{array}{l} \mathsf{pro} : (\Gamma \to ((\alpha, \beta) \times (\beta \to \mathtt{H}))) \to \Gamma \overset{\alpha}{\Rightarrow} \mathtt{H} \\ [\![\mathsf{pro}]\!] \coloneqq \lambda n.\lambda g.\,\{\langle \mathsf{var}\,n\,g, \mathsf{mod}\,n\,\mathsf{snd}\,g\rangle\} \end{array}$$

The argument to this pronoun is a lens built from Z, L, and R, and until Section 6.1, we will assume it is in fact just one of the Z and R lenses spelled out in (74). With such a lens in hand, the pronoun in (75) will do two things. First, it locates an element α from some position in the state Γ, and returns that α as its computed referent. Second, it passes on a modified state $\mathtt{H}$ in which α has been removed. In short, a pronoun pops its referent off the stack for use in local composition. For instance:

(76) $[\![\mathrm{pro}\,\mathsf{Z}]\!] = \lambda(a,g).\{\langle a,g\rangle\}$
$[\![\mathrm{pro}\,(\mathsf{R}\,\mathsf{Z})]\!] = \lambda(x,(a,g)).\{\langle a,(x,g)\rangle\}$
$[\![\mathrm{pro}\,(\mathsf{R}\,(\mathsf{R}\,\mathsf{Z}))]\!] = \lambda(x,(y,(a,g))).\{\langle a,(x,(y,g))\rangle\}$
$[\![\mathrm{pro}\,(\mathsf{R}\,(\mathsf{R}\,(\mathsf{R}\,\mathsf{Z})))]\!] = \lambda(x,(y,(z,(a,g)))).\{\langle a,(x,(y,(z,g)))\rangle\}$
...

There are two reasons for having the pronoun remove its referent from the context. One is that it guarantees paycheck pronouns (and the other sorts of sloppy derivations discussed in Section 7) will be well-defined. No two pronouns can access the same state. Whichever one goes first will change the state that the second one sees. In particular, the sorts of self-referential operations identified by Muskens (1995) cannot arise. When a pronominal computation projects a value v out of a context g, the context is immediately changed; in particular, v is removed from g. So even if v is itself a pronominal computation, the new context that it is evaluated in will not contain v anymore, so circularity is impossible.

The other reason is to keep the fragment as close to static presentations of variable-free semantics as possible. Recall that in frameworks like those of Szabolcsi (1989), Jacobson (1999), and Shan & Barker (2006), pronominal dependencies are, semantically, just functional dependencies, and binding is just a matter of passing a value *as an argument* to a bindee. But then, there is a clear sense in which each "referent" that the grammar conjures up can only value a single pronoun, namely, the one whose open argument it saturates. Even so, there are still two ways that a single binder may come to co-vary with multiple pronouns. First, one pronoun may bind another. Second, since $\triangleright$ is a fully polymorphic, productive operation, it may apply more than once to a single expression. In terms of mechanics, these are very similar, as shown by the parallel derivations in (77).

Again for readability, we abbreviate indices with numerals, such that $\mathrm{pro}_k \equiv \mathrm{pro}\,(\underbrace{\mathsf{R}\,(\ldots(\mathsf{R}}_{k\text{-times}}\,\mathsf{Z})\ldots))$.

(77)
$$\dfrac{\dfrac{\dfrac{}{\mathrm{pro}_0 : (\mathsf{e},\Gamma) \overset{e}{\Rightarrow} \Gamma}[z]}{\mathrm{pro}_0 : \dfrac{(\mathsf{e},\Gamma) \overset{\tau}{\Rightarrow} \mathtt{H} \,\big|\, \Gamma \overset{\tau}{\Rightarrow} \mathtt{H}}{\mathsf{e}}}[\star]}{(\mathrm{pro}_0)^{\triangleright} : \dfrac{(\mathsf{e},\Gamma) \overset{\tau}{\Rightarrow} \mathtt{H} \,\big|\, \Gamma \overset{\tau}{\Rightarrow} \mathtt{H}}{\mathtt{I} \overset{e}{\Rightarrow} (\mathsf{e},\mathtt{\Theta})}}[\text{K-}\triangleright]} \qquad \dfrac{\dfrac{\dfrac{}{\mathrm{Mary}^{\triangleright} : \Gamma \overset{e}{\Rightarrow} (\mathsf{e},\Gamma)}[\text{Lex},\triangleright]}{\mathrm{Mary}^{\triangleright} : \dfrac{\Gamma \overset{\tau}{\Rightarrow} \mathtt{H} \,\big|\, (\mathsf{e},\Gamma) \overset{\tau}{\Rightarrow} \mathtt{H}}{\mathsf{e}}}[\star]}{(\mathrm{Mary}^{\triangleright})^{\triangleright} : \dfrac{\Gamma \overset{\tau}{\Rightarrow} \mathtt{H} \,\big|\, (\mathsf{e},\Gamma) \overset{\tau}{\Rightarrow} \mathtt{H}}{\mathtt{I} \overset{e}{\Rightarrow} (\mathsf{e},\mathtt{\Theta})}}[\text{K-}\triangleright]}$$

With these ingredients, we can put together basic binding derivations as in (78):

(78a)
$$\dfrac{\dfrac{\dfrac{}{\text{called} : \mathsf{e}\to\mathsf{e}\to\mathsf{t}}\,[\text{Lex}]}{\text{called} : \dfrac{\rho \,\big|\, \rho}{\mathsf{e}\to\mathsf{e}\to\mathsf{t}}}\,[\mathbf{U}] \qquad \dfrac{\dfrac{}{\text{pro}_0 : (\mathsf{e},\Gamma) \overset{\mathsf{e}}{\Rightarrow} \Gamma}\,[z]}{\text{pro}_0 : \dfrac{(\mathsf{e},\Gamma) \overset{\tau}{\Rightarrow} \mathsf{H} \,\big|\, \Gamma \overset{\tau}{\Rightarrow} \mathsf{H}}{\mathsf{e}}}\,[\star]}{\text{called pro}_0 : \dfrac{(\mathsf{e},\Gamma) \overset{\tau}{\Rightarrow} \mathsf{H} \,\big|\, \Gamma \overset{\tau}{\Rightarrow} \mathsf{H}}{\mathsf{e}\to\mathsf{t}}}\,[\text{K-/}]$$

(78b)
$$\dfrac{\dfrac{\dfrac{}{\text{someone's} : \Gamma \overset{\mathsf{e}}{\Rightarrow} (\mathsf{e},\Gamma)}\,[\text{Lex}]}{\text{so's} : \dfrac{\Gamma \overset{\tau}{\Rightarrow} \mathsf{H} \,\big|\, (\mathsf{e},\Gamma) \overset{\tau}{\Rightarrow} \mathsf{H}}{\mathsf{e}}}\,[\star] \qquad \dfrac{\dfrac{}{\text{mom} : \mathsf{e}\to\mathsf{e}}\,[\text{Lex}]}{\text{mom} : \dfrac{\rho \,\big|\, \rho}{\mathsf{e}\to\mathsf{e}}}\,[\mathbf{U}]}{\text{so's mom} : \dfrac{\Gamma \overset{\tau}{\Rightarrow} \mathsf{H} \,\big|\, (\mathsf{e},\Gamma) \overset{\tau}{\Rightarrow} \mathsf{H}}{\mathsf{e}}}\,[\text{K-\textbackslash}] \qquad \begin{array}{c}(78a)\\ \vdots \\ \text{called pro}_0 : \dfrac{(\mathsf{e},\Gamma) \overset{\tau}{\Rightarrow} \mathsf{H} \,\big|\, \Gamma \overset{\tau}{\Rightarrow} \mathsf{H}}{\mathsf{e}\to\mathsf{t}}\end{array}$$

$$\dfrac{\text{so's mom called pro}_0 : \dfrac{\Gamma \overset{\tau}{\Rightarrow} \mathsf{H} \,\big|\, \Gamma \overset{\tau}{\Rightarrow} \mathsf{H}}{\mathsf{t}}\,[\text{K-\textbackslash}]}{\text{so's mom called pro}_0 : \dfrac{\Gamma \overset{\tau}{\Rightarrow} \mathsf{H} \,\big|\, \Gamma \overset{\tau}{\Rightarrow} \mathsf{H}}{\overset{\mathsf{t}}{\sigma \Rightarrow \sigma}}}\,[\text{K-}\eta]$$

(78c) $[\![(78\text{b})]\!] = \lambda k.\lambda g.\ \bigcup\{k\,(\lambda i.\,\{\langle \mathbf{call}\,x\,(\mathbf{mom}\,x), i\rangle\})\,g \mid x \in D\}$

The final continuation passed into the denotation of (78b) will be determined by the lowering operation **D**. For instance, if we were to evaluate $[\![(78\text{b})]\!]$ at the identity function, we would get the update $\lambda g.\,\{\langle \mathbf{call}\,x\,(\mathbf{mom}\,x),\, g\rangle \mid x \in D\}$. At any input context g, this update would return, for each individual x, the proposition that x's mother called x, together with the output context g. Assuming, as is standard, that such an update is true if *any* of its witnesses is true, this is exactly as desired.

But just as in Section 4, choosing to pass an identity function as the final step of evaluation would render it impossible to distinguish between (78b) and (79).

(79) *Their mother called someone

In both cases, we would have an indefinite binding a pronoun in virtue of outscoping it, and thereby augmenting the context in which the pronoun is evaluated. So the final ingredient of our analysis, to which we turn in the next section, is an adaptation of **D** to the dynamic setting.

6 Crossover

In Section 4, we showed that a straightforward way to guarantee that scope does not feed backward binding is to completely wipe the input context before executing the underlying meaning. The rule in (80) pursues the same idea.

(80)
$$\left[\!\!\left[\dfrac{E : \dfrac{\rho \,\big|\, \Gamma \overset{\sigma}{\Rightarrow} (\mathsf{H},\Gamma)}{() \overset{\sigma}{\Rightarrow} \mathsf{H}}}{E : \rho}\,[\mathbf{D}]\right]\!\!\right] \coloneqq [\![E]\!]\,(\lambda m.\lambda g.\,\{\langle x,(h,g)\rangle \mid \langle x,h\rangle \in m\,()\})$$

This rule is more similar to the static lowering semantics in Section 3 than it might seem at first blush. If the rule were written as in (81) instead, it would in fact be equivalent, since $\{\langle x, h\rangle \mid \langle x, h\rangle \in m()\} = m()$.

(81) $$\left[\!\!\left[\dfrac{E : \dfrac{\rho \;\big|\; \Gamma \overset{\sigma}{\Rightarrow} (\mathrm{H}, \Gamma)}{() \overset{\sigma}{\Rightarrow} \mathrm{H}}}{E : \rho}\,[\mathbf{D}] \right]\!\!\right] \coloneqq [\![E]\!]\,(\lambda m.\lambda g.\,\{\langle x, h\rangle \mid \langle x, h\rangle \in m()\})$$

The only difference is that in this setting, where modifications to the context can have long-term effects on the discourse, we do not want to completely throw away the incoming state g. Downstream pronouns may yet wish to reference it. So we do the obvious thing and append it behind the new referents introduced by m, which are stored in its output h.

Returning to (78b), notice that the underlying type at the final stage of the derivation is completely polymorphic in its input: $\sigma \overset{t}{\Rightarrow} \sigma$. This means that in particular, it is compatible with the concrete type $() \overset{t}{\Rightarrow} ()$. This type can feed the **D** rule defined in (80), completing the derivation as in (82a):

(78b)

$$\vdots$$

(82a) $$\dfrac{\text{so's mom called pro}_0 : \dfrac{\Gamma \overset{\tau}{\Rightarrow} \mathrm{H} \;\big|\; \Gamma \overset{\tau}{\Rightarrow} \mathrm{H}}{() \overset{t}{\Rightarrow} ()}}{\text{so's mom called pro}_0 : \Gamma \overset{t}{\Rightarrow} ((), \Gamma)}\,[\mathbf{D}]$$

(82b) $$[\![(82a)]\!] = \lambda g.\,\{\mathbf{call}\,x\,(\mathbf{mom}\,x)\,((), g) \mid x \in D\}$$

On the other hand, crossover configurations inevitably run afoul of the type requirements for **D**. For instance, the derivation in (83) goes as far as it can in giving 'someone' inverse scope over a pronoun that it would bind. But this binding will never succeed because the would-be nuclear scope of the indefinite does not have an appropriate type.

(83)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\text{pro}_0\text{'s} : (\mathsf{e},\Gamma) \overset{\mathsf{e}}{\Rightarrow} \Gamma}[z] \quad \dfrac{}{\text{mom} : \mathsf{e} \to \mathsf{e}}[\text{Lex}]}{\text{pro}_0\text{'s} : \dfrac{(\mathsf{e},\Gamma) \overset{\tau}{\Rightarrow} \mathsf{H} \mid \Gamma \overset{\tau}{\Rightarrow} \mathsf{H}}{\mathsf{e}}}[\star] \quad \text{mom} : \dfrac{\rho \mid \rho}{\mathsf{e} \to \mathsf{e}}}[\mathbf{U}]}{\text{pro}_0\text{'s mom} : \dfrac{(\mathsf{e},\Gamma) \overset{\tau}{\Rightarrow} \mathsf{H} \mid \Gamma \overset{\tau}{\Rightarrow} \mathsf{H}}{\mathsf{e}}}[\text{K-}\backslash] \quad \text{called someone} : \dfrac{\Gamma' \overset{\tau'}{\Rightarrow} \mathsf{H}' \mid (\mathsf{e},\Gamma') \overset{\tau'}{\Rightarrow} \mathsf{H}'}{\mathsf{e} \to \mathsf{t}}}[\text{K-}\mathbf{U}]}{\text{pro}_0\text{'s mom} : \dfrac{\dfrac{\rho \mid \rho}{(\mathsf{e},\Gamma) \overset{\tau}{\Rightarrow} \mathsf{H} \mid \Gamma \overset{\tau}{\Rightarrow} \mathsf{H}}}{\mathsf{e}} \quad \text{called someone} : \dfrac{\dfrac{\Gamma' \overset{\tau'}{\Rightarrow} \mathsf{H}' \mid (\mathsf{e},\Gamma') \overset{\tau'}{\Rightarrow} \mathsf{H}'}{\rho' \mid \rho'}}{\mathsf{e} \to \mathsf{t}}}[\text{K}^2\text{-}\backslash]$$

$$\dfrac{\text{pro}_0\text{'s mom called so} : \dfrac{\dfrac{\Gamma' \overset{\tau'}{\Rightarrow} \mathsf{H}' \mid (\mathsf{e},\Gamma') \overset{\tau'}{\Rightarrow} \mathsf{H}'}{(\mathsf{e},\Gamma) \overset{\tau}{\Rightarrow} \mathsf{H} \mid \Gamma \overset{\tau}{\Rightarrow} \mathsf{H}}}{\mathsf{t}}}[\text{K}^2\text{-}\eta]$$

$$\dfrac{\text{pro}_0\text{'s mom called so} : \dfrac{\dfrac{\Gamma' \overset{\tau'}{\Rightarrow} \mathsf{H}' \mid (\mathsf{e},\Gamma') \overset{\tau'}{\Rightarrow} \mathsf{H}'}{(\mathsf{e},\Gamma) \overset{\tau}{\Rightarrow} \mathsf{H} \mid \Gamma \overset{\tau}{\Rightarrow} \mathsf{H}}}{() \overset{\mathsf{t}}{\Rightarrow} ()}}[\text{K-}\mathbf{D}]$$

$$\text{pro}_0\text{'s mom called so} : \dfrac{\Gamma' \overset{\tau'}{\Rightarrow} \mathsf{H}' \mid (\mathsf{e},\Gamma') \overset{\tau'}{\Rightarrow} \mathsf{H}'}{(\mathsf{e},\Gamma) \overset{\mathsf{t}}{\Rightarrow} ((),\Gamma)}$$

The effective precedence predictions for binding extend immediately to secondary crossover. Without belaboring the details of donkey anaphora, assume the denotation of a complex DP like 'every dog with a toy' is as in (84). The crucial thing to note is that the scope of the quantifier will be evaluated in a context that begins with referents for the dog and the toy.

(84) $\text{every dog with a toy} : \dfrac{\Gamma \overset{\mathsf{t}}{\Rightarrow} \Gamma \mid (\mathsf{e},(\mathsf{e},\Gamma)) \overset{\mathsf{t}}{\Rightarrow} \mathsf{H}}{\mathsf{e}}$

$[\![\text{every dog with a toy}]\!] = \lambda k.\lambda g.\{\langle \forall x \forall y.\, \mathbf{dog}\, x \wedge \mathbf{toy}\, y \wedge \mathbf{with}\, y\, x \to k\, x\, (x,(y,g)),g\rangle\}$

The verb phrase 'buried pro_1' demands as input a context containing at least two elements, the second of which is an entity. Since this is exactly the sort of context that the universal to its left provides, composition is seamless. The referent passed in by the quantifier values the pronoun, and the relevant truth conditions are returned.

(85a)

$$\dfrac{\dfrac{\dfrac{\text{every dog with a toy} : \dfrac{\Gamma \overset{\mathsf{t}}{\Rightarrow} \Gamma \mid (\mathsf{e},(\mathsf{e},\Gamma)) \overset{\mathsf{t}}{\Rightarrow} \mathsf{H}}{\mathsf{e}} \quad \text{buried pro}_1 : \dfrac{(\alpha,(\mathsf{e},\Gamma)) \overset{\tau}{\Rightarrow} \mathsf{H} \mid (\alpha,\Gamma) \overset{\tau}{\Rightarrow} \mathsf{H}}{\mathsf{e} \to \mathsf{t}}}[\text{K-}\backslash]}{\text{every dog with a toy buried pro}_1 : \dfrac{\Gamma \overset{\mathsf{t}}{\Rightarrow} \Gamma \mid (\mathsf{e},\Gamma) \overset{\mathsf{t}}{\Rightarrow} \mathsf{H}}{\mathsf{t}}}[\text{K-}\eta]}{\text{every dog with a toy buried pro}_1 : \dfrac{\Gamma \overset{\mathsf{t}}{\Rightarrow} \Gamma \mid (\mathsf{e},\Gamma) \overset{\mathsf{t}}{\Rightarrow} \mathsf{H}}{() \overset{\mathsf{t}}{\Rightarrow} ()}}[\mathbf{D}]$$

$$\text{every dog with a toy buried pro}_1 : \Gamma \overset{\mathsf{t}}{\Rightarrow} \Gamma$$

(85b) $[\![(85a)]\!] = \lambda g.\{\langle \forall x \forall y.\, \mathbf{dog}\, x \wedge \mathbf{toy}\, y \wedge \mathbf{with}\, y\, x \rightarrow \mathbf{bury}\, y\, x, g\rangle\}$

An attempt at binding from right to left would require inverse scope. The problem here is no different than in the primary crossover configuration: lowering is ill-typed. It makes no difference that the offending binding would come from a nested, "donkey" indefinite. The issue is simply that constituents with unbound pronouns do not have the right types to serve as nuclear scopes.

$$\dfrac{\dfrac{\text{pro}_1 : \dfrac{(\alpha, (\mathtt{e}, \Gamma')) \overset{\tau}{\Rightarrow} \mathtt{H}' \mid (\alpha, \Gamma') \overset{\tau}{\Rightarrow} \mathtt{H}'}{\mathtt{e}} \qquad \text{delighted every dog with a toy} : \dfrac{\Gamma \overset{t}{\Rightarrow} \Gamma \mid (\mathtt{e}, (\mathtt{e}, \Gamma)) \overset{t}{\Rightarrow} \mathtt{H}}{\mathtt{e} \rightarrow \mathtt{t}}}{\text{pro}_1 \text{ delighted every dog with a toy} : \dfrac{\dfrac{\rho' \mid \rho'}{(\alpha, (\mathtt{e}, \Gamma')) \overset{\tau}{\Rightarrow} \mathtt{H}' \mid (\alpha, \Gamma') \overset{\tau}{\Rightarrow} \mathtt{H}'}}{\mathtt{e}}}\,[\mathbf{U}] \qquad \dfrac{\Gamma \overset{t}{\Rightarrow} \Gamma \mid (\mathtt{e}, (\mathtt{e}, \Gamma)) \overset{t}{\Rightarrow} \mathtt{H}}{\dfrac{\rho \mid \rho}{\mathtt{e} \rightarrow \mathtt{t}}}}{\text{pro}_1 \text{ delighted every dog with a toy} : \dfrac{\Gamma \overset{t}{\Rightarrow} \Gamma \mid (\mathtt{e}, (\mathtt{e}, \Gamma)) \overset{t}{\Rightarrow} \mathtt{H}}{(\alpha, (\mathtt{e}, \Gamma')) \overset{t}{\Rightarrow} ((), (\alpha, \Gamma'))}}\,[\text{K-}\backslash]\,[\text{K-}\mathbf{U}]$$

(86)

$$\dfrac{\dfrac{\text{pro}_1 \text{ delighted every dog with a toy} : \dfrac{\Gamma \overset{t}{\Rightarrow} \Gamma \mid (\mathtt{e}, (\mathtt{e}, \Gamma)) \overset{t}{\Rightarrow} \mathtt{H}}{(\alpha, (\mathtt{e}, \Gamma')) \overset{\tau}{\Rightarrow} \mathtt{H}' \mid (\alpha, \Gamma') \overset{\tau}{\Rightarrow} \mathtt{H}'}}{\mathtt{t}}}{\text{pro}_1 \text{ delighted every dog with a toy} : \dfrac{\dfrac{\Gamma \overset{t}{\Rightarrow} \Gamma \mid (\mathtt{e}, (\mathtt{e}, \Gamma)) \overset{t}{\Rightarrow} \mathtt{H}}{(\alpha, (\mathtt{e}, \Gamma')) \overset{\tau}{\Rightarrow} \mathtt{H}' \mid (\alpha, \Gamma') \overset{\tau}{\Rightarrow} \mathtt{H}'}}{() \overset{t}{\Rightarrow} ()}}\,[\mathrm{K}^2\text{-}\mathbf{D}]\,[\mathrm{K}^2\text{-}\eta]$$

6.1 Gorn indices

Look again at the lowering rule in (80). As described above, the outputs of the nuclear scope H are added to the input context Γ in one whole chunk. This is done largely because it is the simplest imaginable way to get them onto the record that gets passed along to subsequent discourse. One consequence of this decision is that the context is no longer strictly right-branching, i.e., list-structured. It is, instead, tree-structured.[2]

2 The context could also easily be linearized right at the point of lowering, so that it always remains right-branching (i.e., list-like). Then this section could be deleted, and the rest of the analysis would carry on unchanged. We just need an operator that concatenates two right-branching contexts. Such an operator is defined recursively in (87), which doubles as both a type- and value-level definition of concatenation. Then the semantics of [**D**] simply swaps out (h, g) for $h^\frown g$, as in (88).

(87) $()^\frown h \coloneqq g$
$(a, b)^\frown g \coloneqq (a, b^\frown g)$

(88) $$\left[\!\!\left[\dfrac{E : \dfrac{\rho \mid \Gamma \overset{\sigma}{\Rightarrow} \mathtt{H}^\frown\Gamma}{() \overset{\sigma}{\Rightarrow} \mathtt{H}}}{E : \rho}\,[\mathbf{D}] \right]\!\!\right] \coloneqq [\![E]\!]\,(\lambda m.\lambda g.\{\langle x, h^\frown g\rangle \mid \langle x, h\rangle \in m()\})$$

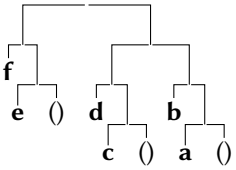
The only hitch is that the concatenation operator $^\frown$ is not technically parametric in its polymorphism. Its type would contain type variables that range over some but not all types (namely, all and only tuple types in its first argument). This is the sort of *ad hoc* polymorphism familiar from Generalized Conjunction. It adds no expressive power to the type system, but does mean that type *constraints* (e.g., Γ is a tuple) need to be tracked in the inference rules. We opt for the branching state instead to sidestep this, but it is completely inconsequential.

Loosely speaking, each evaluation domain will start its own left branch. For instance, the discourse in (89a) might naturally emerge as in (89b), drawn more graphically in (89c), assuming each clause corresponds to a single evaluation domain, or scope island.

(89a) 'A' precedes 'B', and 'C' precedes 'D', and 'E' precedes 'F'.

(89b) $((\mathbf{f}, (\mathbf{e}, ())), ((\mathbf{d}, (\mathbf{c}, ())), (\mathbf{b}, (\mathbf{a}, ()))))$

(89c)



That said, the picture here, though illustrative of the canonical effect, should be taken with a grain of salt. The exact location of a referent in the context will depend on the fine details of the derivation, since those details are determined partly by scope and evaluation order, which can vary "spuriously" (without truth-conditional consequences) across derivations. For instance, there is an even simpler derivation of (89a) in which the referents do in fact emerge in a descending list, despite forced evaluation at each clause boundary.

To complete the semantics, we simply allow indices to range over any well-typed combination of L, R, and Z. Since L and R are lens modifiers, and Z is a lens, it is clear such well-typed combinations will always be of the form $l_1\,(l_2\,(l_3\,(l_4 \ldots \mathsf{Z})))$, where each l_i is either L or R. In other words, an index is effectively a string of L's and R's, concluded by a Z. Such strings may quite literally be read as Gorn addresses for the nodes of a tree, where L and Z mean go Left, and R means go Right.

For instance, in order to grab **e** from the state in (89c), starting from the root, you'd need to go Left, then Right, then Left. And indeed, the pronoun 'pro $(\mathsf{L}\,(\mathsf{R}\,\mathsf{Z}))$' : $((\varphi, (\varepsilon, \theta)), \Gamma) \overset{\varepsilon}{\Rightarrow} ((\varphi, \theta), \Gamma)$ will grab, and remove, **e** from (89c). To target **c** instead, you'd need to go Right, then Left, then Right, then Left. The corresponding pronoun 'pro $(\mathsf{R}\,(\mathsf{L}\,(\mathsf{R}\,\mathsf{Z})))$' : $(\Gamma, ((\delta, (\varsigma, \theta)), \mathtt{H})) \overset{\varsigma}{\Rightarrow} (\Gamma, ((\delta, \theta), \mathtt{H}))$ will do just that: extract and eliminate ς from the state in (89c). And in general, every sequence of L's and R's will correspond to a node in some tree of referents, and every node but one in such a tree will correspond to some sequence of L's and R's. The only inaccessible coordinate is the very rightmost leaf (since all indices end in Z, which is a step to the left). But no discourse antecedent can ever end up in that position because pushing a value with $\triangleright$ always *prepends* that value to a context. This guarantees that any remembered referent is always to the left of something.

We will not make any interesting use of the tree-structured state, and in the remainder, we continue to use numeral abbreviations on pronouns: $\mathrm{pro}_k \equiv \mathrm{pro}\,(\mathsf{R}\,(\ldots(\mathsf{R}\,\mathsf{Z})\ldots))$, with k-many R's.

7 Cross-categorial anaphora

One of the motivations for the framework developed here is the patent *prima facie* need for discourse referents of different types. But so far all of the examples have stuck to plain entities. Generalizing these examples to other kinds of first-order objects requires no ingenuity, thanks to the polymorphism of the interpretation rules. The example in (90), for instance, derives ordinary VP-ellipsis. Here, the ellipsis gap is treated as a pronoun with exactly the same semantics as in all the examples so far. For clarity, since this is a pro-VP rather than a pronoun, we write did_0 instead of pro_0.

(90a)
$$\dfrac{\dfrac{\dfrac{}{\text{Mary} : \mathsf{e}}\,[\text{Lex}]}{\text{Mary} : \dfrac{\rho \,\big|\, \rho}{\mathsf{e}}}\,[\mathbf{U}] \qquad \dfrac{\dfrac{\dfrac{}{\text{called} : \mathsf{e} \to \mathsf{t}}\,[\text{Lex}]}{\text{called} : \Gamma \overset{\mathsf{e}\to\mathsf{t}}{\Rightarrow} (\mathsf{e} \to \mathsf{t}, \Gamma)}\,[\triangleright]}{\text{called} : \dfrac{\Gamma \overset{\tau}{\Rightarrow} \mathsf{H} \,\big|\, (\mathsf{e} \to \mathsf{t}, \Gamma) \overset{\tau}{\Rightarrow} \mathsf{H}}{\mathsf{e} \to \mathsf{t}}}\,[\star]}{\text{Mary called} : \dfrac{\Gamma \overset{\tau}{\Rightarrow} \mathsf{H} \,\big|\, (\mathsf{e} \to \mathsf{t}, \Gamma) \overset{\tau}{\Rightarrow} \mathsf{H}}{\mathsf{t}}}\,[\text{K-}\backslash]$$

(90b) $[\![(90a)]\!] = \lambda k.\lambda g.\ \bigcup\{k\,(\mathbf{call}\,\mathbf{m})\,(\mathbf{call}, g)\}$

(90c)
$$\dfrac{\dfrac{\dfrac{}{\text{John} : \mathsf{e}}\,[\text{Lex}]}{\text{John} : \dfrac{\rho \,\big|\, \rho}{\mathsf{e}}}\,[\mathbf{U}] \qquad \dfrac{\dfrac{}{\text{did}_0 : (\mathsf{e} \to \mathsf{t}, \Gamma) \overset{\mathsf{e}\to\mathsf{t}}{\Rightarrow} \Gamma}\,[z]}{\text{did}_0 : \dfrac{(\mathsf{e} \to \mathsf{t}, \Gamma) \overset{\tau}{\Rightarrow} \mathsf{H} \,\big|\, \Gamma \overset{\tau}{\Rightarrow} \mathsf{H}}{\mathsf{e} \to \mathsf{t}}}\,[\star]}{\text{John did}_0 : \dfrac{(\mathsf{e} \to \mathsf{t}, \Gamma) \overset{\tau}{\Rightarrow} \mathsf{H} \,\big|\, \Gamma \overset{\tau}{\Rightarrow} \mathsf{H}}{\mathsf{t}}}\,[\text{K-}\backslash]$$

(90d) $[\![(90c)]\!] = \lambda k.\lambda (P, g).\ \bigcup\{k\,(P\,\mathbf{j})\,g\}$

(90e)
$$\dfrac{\dfrac{\begin{array}{c}(90a)\\ \vdots \\ \text{Mary called} : \dfrac{\Gamma \overset{\tau}{\Rightarrow} \mathsf{H} \,\big|\, (\mathsf{e} \to \mathsf{t}, \Gamma) \overset{\tau}{\Rightarrow} \mathsf{H}}{\mathsf{t}}\end{array} \qquad \dfrac{\dfrac{}{\text{and} : \mathsf{t} \to \mathsf{t} \to \mathsf{t}}\,[\text{Lex}]}{\text{and} : \dfrac{\rho \,\big|\, \rho}{\mathsf{t} \to \mathsf{t} \to \mathsf{t}}}\,[\mathbf{U}] \qquad \begin{array}{c}(90c)\\ \vdots \\ \text{John did}_0 : \dfrac{(\mathsf{e} \to \mathsf{t}, \Gamma) \overset{\tau}{\Rightarrow} \mathsf{H} \,\big|\, \Gamma \overset{\tau}{\Rightarrow} \mathsf{H}}{\mathsf{t}}\end{array}}{\text{and John did}_0 : \dfrac{(\mathsf{e} \to \mathsf{t}, \Gamma) \overset{\tau}{\Rightarrow} \mathsf{H} \,\big|\, \Gamma \overset{\tau}{\Rightarrow} \mathsf{H}}{\mathsf{t} \to \mathsf{t}}}\,[\text{K-}/]}{\text{Mary called and John did}_0 : \dfrac{(\mathsf{e} \to \mathsf{t}, \Gamma) \overset{\tau}{\Rightarrow} \mathsf{H} \,\big|\, \Gamma \overset{\tau}{\Rightarrow} \mathsf{H}}{\mathsf{t}}}\,[\text{K-}\backslash]$$
$$\dfrac{\text{Mary called and John pro}_0 : \dfrac{(\mathsf{e} \to \mathsf{t}, \Gamma) \overset{\tau}{\Rightarrow} \mathsf{H} \,\big|\, \Gamma \overset{\tau}{\Rightarrow} \mathsf{H}}{() \overset{\mathsf{t}}{\Rightarrow} ()}}{\text{Mary called and John pro}_0 : (\mathsf{e} \to \mathsf{t}, \Gamma) \overset{\mathsf{t}}{\Rightarrow} ((), \Gamma)}\,[\mathbf{D}]\,[\text{K-}\eta]$$

(90f) $[\![(90e)]\!] = \lambda g.\,\{\langle \mathbf{call}\,\mathbf{m} \wedge \mathbf{call}\,\mathbf{j}, ((), g)\rangle\}$

7.1 Higher-order anaphora

More interesting are cases of higher-order anaphora, including sloppy binding, paycheck pronouns, and verb phrase ellipsis with inversely scoping elements. The key to all of these cases is that the [▷] operation is entirely unconstrained in what kinds of referents it can add to the discourse context. In particular, it can add denotations that are themselves dynamic. When these updates are retrieved and executed, the anaphoric elements inside will be executed in a different environment than when they were stored.

To illustrate this, assume that a pronoun-containing VP like 'called her mom' can be derived at type $\pi \equiv (\mathsf{e}, \mathtt{I}) \overset{\mathsf{e}\to\mathsf{t}}{\Rightarrow} \mathtt{I}$. This is the type of an update that pops an entity off the discourse state in determining a particular property, e.g., the property of calling that entity's mom. Since this context-

sensitive property is the meaning that a sloppy understanding of verb phrase ellipsis invokes, we push this entire update onto the context.

(91a)
$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\text{called pro}_0\text{'s mom} : \pi}}{(\text{called pro}_0\text{'s mom})^{\triangleright} : \Gamma \overset{\pi}{\Rightarrow} (\pi, \Gamma)}[\triangleright]}{(\text{called pro}_0\text{'s mom})^{\triangleright} : \dfrac{\Gamma \overset{\tau}{\Rightarrow} \mathtt{H} \mid (\pi, \Gamma) \overset{\tau}{\Rightarrow} \mathtt{H}}{\pi}}[\star]}{(\text{called pro}_0\text{'s mom})^{\triangleright} : \dfrac{\Gamma \overset{\tau}{\Rightarrow} \mathtt{H} \mid (\pi, \Gamma) \overset{\tau}{\Rightarrow} \mathtt{H}}{\dfrac{(\mathtt{e}, \mathtt{I}) \overset{\tau'}{\Rightarrow} \theta \mid \mathtt{I} \overset{\tau'}{\Rightarrow} \theta}{\mathtt{e} \to \mathtt{t}}}}[\text{K-}\star]$$

(91b) $[\![(91a)]\!] = \lambda k.\lambda g.\, k\,(\lambda c.\lambda(x, i).\, c\,(\mathbf{call}\,(\mathbf{mom}\,x))\,i)\,(\mathbf{P}, g)$
where $\mathbf{P} = [\![\text{called pro}_0\text{'s mom}]\!] = \lambda(x, i).\, \{\langle \mathbf{call}\,(\mathbf{mom}\,x), i\rangle\}$

The derivation in (91a) yields an expression whose underlying type is $\mathtt{e} \to \mathtt{t}$, so it can compose where ordinary properites are expected. But it also performs two actions in the computation of that property. The outer action, documented on the top level of the type's tower, adds the function $\mathbf{P} = [\![\text{called pro}_0\text{'s mom}]\!]$ to the discourse state. The inner action, documented in the middle layer of the type's tower, executes $\mathbf{P}$ in a context containing a referent for the pronoun pro_0, just as it would in isolation. This referent may indeed be provided by the subject of the VP, as in (92a).

(92a)
$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\text{Mary} : \mathtt{e}}[\text{Lex}]}{\text{Mary}^{\triangleright} : \mathtt{I} \overset{\mathtt{e}}{\Rightarrow} (\mathtt{e}, \mathtt{I})}[\triangleright]}{\text{Mary}^{\triangleright} : \dfrac{\mathtt{I} \overset{\tau}{\Rightarrow} \theta \mid (\mathtt{e}, \mathtt{I}) \overset{\tau}{\Rightarrow} \theta}{\mathtt{e}}}[\star]}{\text{Mary}^{\triangleright} : \dfrac{\rho \mid \rho}{\dfrac{\mathtt{I} \overset{\tau}{\Rightarrow} \theta \mid (\mathtt{e}, \mathtt{I}) \overset{\tau}{\Rightarrow} \theta}{\mathtt{e}}}}[\mathbf{U}] \qquad (\text{called pro}_0\text{'s mom})^{\triangleright} : \dfrac{\Gamma \overset{\tau}{\Rightarrow} \mathtt{H} \mid (\pi, \Gamma) \overset{\tau}{\Rightarrow} \mathtt{H}}{\dfrac{(\mathtt{e}, \mathtt{I}) \overset{\tau'}{\Rightarrow} \theta \mid \mathtt{I} \overset{\tau'}{\Rightarrow} \theta}{\mathtt{e} \to \mathtt{t}}}}[\text{K}^2\text{-}\backslash]$$
$$\text{Mary}^{\triangleright}\ (\text{called pro}_0\text{'s mom})^{\triangleright} : \dfrac{\Gamma \overset{\tau}{\Rightarrow} \mathtt{H} \mid (\pi, \Gamma) \overset{\tau}{\Rightarrow} \mathtt{H}}{\dfrac{\mathtt{I} \overset{\tau}{\Rightarrow} \theta \mid \mathtt{I} \overset{\tau}{\Rightarrow} \theta}{\mathtt{t}}}$$

There are two things to note about the final type in (92a). First, the outermost layer indicates that the dynamic meaning of the VP will be added to the input context. Second, the middle layer reveals that binding from the subject to the object was successful. In particular, the VP's middle-layer dependence on an input of type $(\mathtt{e}, \mathtt{I})$ has been eliminated, so that the input is now completely free. This is because the subject's output is guaranteed to be of the form $(\mathtt{e}, \mathtt{I})$, no matter the input $\mathtt{I}$, since it will add a referent for Mary to the front of the state.

Composition of a subsequent clause containing an elliptical VP is structurally similar to the composition of (92a). The only thing notable about it is that in order for the composition to work

out, the ellipsis site's polymorphic type will be resolved so that it is expecting the first coordinate of its input context to contain a denotation that is itself dynamic (type $\pi \equiv (e, I) \overset{e \to t}{\Rightarrow} I$). This of course will end up being the dynamic VP that the previous clause adds. Again for clarity we write pro_0 as 'did_0' in the second clause.

(93a)
$$\dfrac{\dfrac{\dfrac{\overline{did_0 : (\pi, \Gamma) \overset{\pi}{\Rightarrow} \Gamma}^{[z]}}{did_0 : \dfrac{(\pi, \Gamma) \overset{\tau}{\Rightarrow} H \mid \Gamma \overset{\tau}{\Rightarrow} H}{\pi}}^{[\star]}}{did_0 : \dfrac{\dfrac{(\pi, \Gamma) \overset{\tau}{\Rightarrow} H \mid \Gamma \overset{\tau}{\Rightarrow} H}{(e, I) \overset{\tau'}{\Rightarrow} \theta \mid I \overset{\tau'}{\Rightarrow} \theta}}{e \to t}}^{[K\text{-}\star]}$$

(93b) $[\![(93a)]\!] = \lambda k.\lambda(\pi, g).k\,(\lambda c.\lambda(x, i).\ \bigcup\{c\,P\,i' \mid \langle P, i'\rangle \in \pi\,(x, i)\})\,g$

Again, because the pro-VP is expecting a dynamic property as its referent, that property will have to be executed in a local context. In particular, since it is a dynamic property of type π, that local context had better contain an entity in its first position. That entity will be popped off the context and used to construct an ordinary property for the second clause. Crucially, the entity provided to the local context of the retrieved VP may contain a new referent pushed onto the stack by the second clause's subject. This is shown in (94a).

(94a)
$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{Sue : e}^{[Lex]}}{Sue^{\triangleright} : I \overset{e}{\Rightarrow} (e, I)}^{[\triangleright]}}{Sue^{\triangleright} : \dfrac{I \overset{\tau}{\Rightarrow} \theta \mid (e, I) \overset{\tau}{\Rightarrow} \theta}{e}}^{[\star]} \qquad \begin{array}{c}(93a)\\ \vdots \\ did_0 : \dfrac{(\pi, \Gamma) \overset{\tau}{\Rightarrow} H \mid \Gamma \overset{\tau}{\Rightarrow} H}{(e, I) \overset{\tau'}{\Rightarrow} \theta \mid I \overset{\tau'}{\Rightarrow} \theta}\end{array}}{Sue^{\triangleright} : \dfrac{\rho \mid \rho}{I \overset{\tau}{\Rightarrow} \theta \mid (e, I) \overset{\tau}{\Rightarrow} \theta}\ \ e}^{[\cup]}}{Sue^{\triangleright}\ did_0 : \dfrac{(\pi, \Gamma) \overset{\tau}{\Rightarrow} H \mid \Gamma \overset{\tau}{\Rightarrow} H}{I \overset{\tau}{\Rightarrow} \theta \mid I \overset{\tau}{\Rightarrow} \theta}\ \ t}^{[K^2\text{-}\backslash]}$$

Conjoining (92a) and (94a) yields the intended effect.

(95a)

$$\dfrac{\text{Mary}^{\triangleright}\ (\text{called pro}_0\text{'s mom})^{\triangleright} : \dfrac{\dfrac{\Gamma \overset{\tau}{\Rightarrow} \mathtt{H} \mid (\pi, \Gamma) \overset{\tau}{\Rightarrow} \mathtt{H}}{\mathtt{I} \overset{\tau}{\Rightarrow} \theta \mid \mathtt{I} \overset{\tau}{\Rightarrow} \theta}}{\mathtt{t}} \quad \text{and} : \dfrac{\dfrac{\dfrac{}{\text{and} : \mathtt{t} \to \mathtt{t} \to \mathtt{t}}\,[\text{Lex}]}{\text{and} : \dfrac{\rho \mid \rho}{\mathtt{t} \to \mathtt{t} \to \mathtt{t}}}\,[\mathbf{U}]}{\dfrac{\rho' \mid \rho'}{\rho \mid \rho}}\,[\mathbf{U}] \quad \text{Sue}^{\triangleright}\ \text{did}_0 : \dfrac{\dfrac{(\pi, \Gamma) \overset{\tau}{\Rightarrow} \mathtt{H} \mid \Gamma \overset{\tau}{\Rightarrow} \mathtt{H}}{\mathtt{I} \overset{\tau}{\Rightarrow} \theta \mid \mathtt{I} \overset{\tau}{\Rightarrow} \theta}}{\mathtt{t}}}{\text{Mary}^{\triangleright}\ (\text{called pro}_0\text{'s mom})^{\triangleright} \text{ and Sue}^{\triangleright}\ \text{did}_0 : \dfrac{\dfrac{\Gamma \overset{\tau}{\Rightarrow} \mathtt{H} \mid \Gamma \overset{\tau}{\Rightarrow} \mathtt{H}}{\mathtt{I} \overset{\tau}{\Rightarrow} \theta \mid \mathtt{I} \overset{\tau}{\Rightarrow} \theta}}{\mathtt{t}}}\,[\text{K}^2\text{-}\backslash, \text{K}^2\text{-}/]$$

(92a) and (94a): ⋮

Here we can see several things. The outer anaphoric effect — the binding of the pro-VP by the antecedent VP — has been successful. The first clause introduced it to the context, and the second clause used it. The inner layer reveals two successful bindings, the pronoun in the antecedent VP by the first subject, Mary, and the pronoun in the recycled VP by the second subject, Sue. The underlying type reveals we have computed a truth value. Lowering twice gives (96b).

(96a)

$$\dfrac{\dfrac{\text{Mary}^{\triangleright}\ (\text{called pro}_0\text{'s mom})^{\triangleright} \text{ and Sue}^{\triangleright}\ \text{did}_0 : \dfrac{\dfrac{\Gamma \overset{\tau}{\Rightarrow} \mathtt{H} \mid \Gamma \overset{\tau}{\Rightarrow} \mathtt{H}}{\mathtt{I} \overset{\tau}{\Rightarrow} \theta \mid \mathtt{I} \overset{\tau}{\Rightarrow} \theta}}{\mathtt{t}}}{\text{Mary}^{\triangleright}\ (\text{called pro}_0\text{'s mom})^{\triangleright} \text{ and Sue}^{\triangleright}\ \text{did}_0 : \dfrac{\dfrac{\Gamma \overset{\tau}{\Rightarrow} \mathtt{H} \mid \Gamma \overset{\tau}{\Rightarrow} \mathtt{H}}{\mathtt{I} \overset{t}{\Rightarrow} ((), \mathtt{I})}}}\,[\text{K-}\mathbf{D}]}{\text{Mary}^{\triangleright}\ (\text{called pro}_0\text{'s mom})^{\triangleright} \text{ and Sue}^{\triangleright}\ \text{did}_0 : \Gamma \overset{t}{\Rightarrow} (((), ()), \Gamma)}\,[\mathbf{D}]$$

(95a): ⋮

(96b) $[\![(96a)]\!] = \lambda g.\{\langle \mathbf{call}\,(\mathbf{mom}\,\mathbf{m})\,\mathbf{m} \wedge \mathbf{call}\,(\mathbf{mom}\,\mathbf{s})\,\mathbf{s}, (((), ()), g)\rangle\}$

The derivation of paycheck binding is almost exactly the same as the sequence of derivations in (92a)–(96a). The only difference in deriving (97), for example, is that instead of pushing the dynamic VP 'spent pro_0's paycheck' to the outer layer, we would push just the dynamic object 'pro_0's paycheck'. And instead of a pro-VP 'did_0' in the second clause, we'd have a pronoun 'pro_0'. But aside from trivially exchanging the underlying types of the pro-forms and antecedents from $\mathtt{e} \to \mathtt{t}$ (in the case of VPE) to $\mathtt{e}$ (in the case of the paycheck pronoun), the dynamic types are identical.

(97) Mary spent her paycheck; Sue saved it

The same holds *mutatis mutandis* for VPs that contain sloppily bound VPs, rather than sloppily bound pronouns, as in (98) (Hardt 1999, Schwarz 2000).

(98) Mary cooks because she likes to; Sue cooks even though she doesn't

Again, the derivation is nearly identical to that of (96a). It's just a matter of swapping in the relevant underlying types.

Inverse scope "out of" an elided VP follows the same pattern. We lean on the polymorphism of context modification to store a quantificational discourse referent. When that referent is recovered,

it is evaluated in an environment where the stored quantificational element has higher precedence than its surroundings. Assume that a quantifier-containing VP like ‘called everyone’ can be derived at type $\kappa \equiv \dfrac{\mathtt{I} \overset{\mathtt{t}}{\Rightarrow} \mathtt{I} \mid \mathtt{I} \overset{\mathtt{t}}{\Rightarrow} \Theta}{\mathtt{e} \to \mathtt{t}}$. Such a VP meaning can be pushed onto the state whole hog.

(99a)

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\text{a nurse} : \mathtt{M} \overset{\mathtt{e}}{\Rightarrow} \mathtt{M}}\,[\text{Lex}]}{\text{a nurse} : \dfrac{\mathtt{M} \overset{\sigma}{\Rightarrow} \mathtt{N} \mid \mathtt{M} \overset{\sigma}{\Rightarrow} \mathtt{N}}{\mathtt{e}}}\,[\star]}{\text{a nurse} : \dfrac{\circ \mid \circ}{\dfrac{\rho \mid \rho}{\dfrac{\mathtt{M} \overset{\sigma}{\Rightarrow} \mathtt{N} \mid \mathtt{M} \overset{\sigma}{\Rightarrow} \mathtt{N}}{\mathtt{e}}}}}\,[\mathbf{U}, \mathbf{U}] \qquad \dfrac{\dfrac{\dfrac{\dfrac{\begin{matrix}\vdots \\ \text{called everyone} : \kappa\end{matrix}}{(\text{called everyone})^{\triangleright} : \Gamma \overset{\kappa}{\Rightarrow} (\kappa, \Gamma)}\,[\triangleright]}{(\text{called everyone})^{\triangleright} : \dfrac{\Gamma \overset{\tau}{\Rightarrow} \mathtt{H} \mid (\kappa, \Gamma) \overset{\tau}{\Rightarrow} \mathtt{H}}{\kappa}}\,[\star]}{(\text{called everyone})^{\triangleright} : \dfrac{\Gamma \overset{\tau}{\Rightarrow} \mathtt{H} \mid (\kappa, \Gamma) \overset{\tau}{\Rightarrow} \mathtt{H}}{\dfrac{\mathtt{I} \overset{\mathtt{t}}{\Rightarrow} \mathtt{I} \mid \mathtt{I} \overset{\mathtt{t}}{\Rightarrow} \Theta}{\mathtt{e} \to \mathtt{t}}}}\,[\equiv]}{(\text{called everyone})^{\triangleright} : \dfrac{\Gamma \overset{\tau}{\Rightarrow} \mathtt{H} \mid (\kappa, \Gamma) \overset{\tau}{\Rightarrow} \mathtt{H}}{\dfrac{\mathtt{I} \overset{\mathtt{t}}{\Rightarrow} \mathtt{I} \mid \mathtt{I} \overset{\mathtt{t}}{\Rightarrow} \Theta}{\dfrac{\chi \mid \chi}{\mathtt{e} \to \mathtt{t}}}}}\,[\mathrm{K}^2\text{-}\mathbf{U}]}{\text{a nurse (called everyone)}^{\triangleright} : \dfrac{\Gamma \overset{\tau}{\Rightarrow} \mathtt{H} \mid (\kappa, \Gamma) \overset{\tau}{\Rightarrow} \mathtt{H}}{\dfrac{\mathtt{I} \overset{\mathtt{t}}{\Rightarrow} \mathtt{I} \mid \mathtt{I} \overset{\mathtt{t}}{\Rightarrow} \Theta}{\dfrac{\mathtt{M} \overset{\sigma}{\Rightarrow} \mathtt{N} \mid \mathtt{M} \overset{\sigma}{\Rightarrow} \mathtt{N}}{\mathtt{t}}}}}\,[\mathrm{K}^3\text{-}\backslash]$$

$$\dfrac{\text{a nurse (called everyone)}^{\triangleright} : \dfrac{\Gamma \overset{\tau}{\Rightarrow} \mathtt{H} \mid (\kappa, \Gamma) \overset{\tau}{\Rightarrow} \mathtt{H}}{\dfrac{\mathtt{I} \overset{\mathtt{t}}{\Rightarrow} \mathtt{I} \mid \mathtt{I} \overset{\mathtt{t}}{\Rightarrow} \Theta}{\dfrac{\mathtt{M} \overset{\sigma}{\Rightarrow} \mathtt{N} \mid \mathtt{M} \overset{\sigma}{\Rightarrow} \mathtt{N}}{\mathtt{t}}}}}{\text{a nurse (called everyone)}^{\triangleright} : \Gamma \overset{\mathtt{t}}{\Rightarrow} (\ldots, (\kappa, \Gamma))}\,[\mathrm{K}^3\text{-}\eta, \mathrm{K}^2\text{-}\mathbf{D}, \mathrm{K}\text{-}\mathbf{D}, \mathbf{D}]$$

(99b) $[\![(99\text{a})]\!] = \lambda g. \{\langle \forall x. \exists y. \mathbf{nurse}\, y \wedge \mathbf{call}\, x\, y, (\ldots, (\mathbf{Q}, g))\rangle\}$

where $\mathbf{Q} = [\![\text{called everyone}]\!] = \lambda k. \lambda i. \{\langle \forall x. \exists j. \langle \mathbf{T}, j\rangle \in k\, (\mathbf{call}\, x)\, i, i\rangle\}$

The significant effect of the derivation in (99a) is to add the denotation of ‘called everyone’ to the outgoing discourse context. The rest of the derivation computes the truth conditions associated with the inverse-scope reading of ‘a nurse called everyone’, as revealed in (99b). As in the matching derivations of clauses in the prior ellipsis examples, a subsequent clause with VPE can be derived so that it is missing exactly the sort of thing that (99a) provides. This is shown in (100a).

(100a)

$$\dfrac{\dfrac{\dfrac{}{\text{a doctor} : \mathtt{M} \overset{\mathtt{e}}{\Rightarrow} \mathtt{M}}\,[\text{Lex}]}{\text{a doctor} : \dfrac{\mathtt{M} \overset{\sigma}{\Rightarrow} \mathtt{N} \,\big|\, \mathtt{M} \overset{\sigma}{\Rightarrow} \mathtt{N}}{\mathtt{e}}}\,[\star] \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\text{did}_1 : (\alpha, (\kappa, \Gamma)) \overset{\kappa}{\Rightarrow} (\alpha, \Gamma)}\,[\text{sz}]}{\text{did}_1 : \dfrac{(\alpha, (\kappa, \Gamma)) \overset{\tau}{\Rightarrow} \mathtt{H} \,\big|\, (\alpha, \Gamma) \overset{\tau}{\Rightarrow} \mathtt{H}}{\kappa}}\,[\star]}{\text{did}_1 : \dfrac{(\alpha, (\kappa, \Gamma)) \overset{\tau}{\Rightarrow} \mathtt{H} \,\big|\, (\alpha, \Gamma) \overset{\tau}{\Rightarrow} \mathtt{H}}{\mathtt{I} \overset{\mathtt{t}}{\Rightarrow} \mathtt{I} \,\big|\, \mathtt{I} \overset{\mathtt{t}}{\Rightarrow} \theta \over \mathtt{e} \to \mathtt{t}}}\,[\equiv]}{\text{a nurse} : \dfrac{\dfrac{\mathtt{o} \,\big|\, \mathtt{o}}{\rho \,\big|\, \rho} \over \mathtt{M} \overset{\sigma}{\Rightarrow} \mathtt{N} \,\big|\, \mathtt{M} \overset{\sigma}{\Rightarrow} \mathtt{N}}{\mathtt{e}}}\,[\mathbf{U}, \mathbf{U}] \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\text{did}_1 : (\alpha, (\kappa, \Gamma)) \overset{\kappa}{\Rightarrow} (\alpha, \Gamma)}\,[\text{sz}]}{\text{did}_1 : \dfrac{(\alpha, (\kappa, \Gamma)) \overset{\tau}{\Rightarrow} \mathtt{H} \,\big|\, (\alpha, \Gamma) \overset{\tau}{\Rightarrow} \mathtt{H}}{\kappa}}\,[\star]}{\text{did}_1 : \dfrac{(\alpha, (\kappa, \Gamma)) \overset{\tau}{\Rightarrow} \mathtt{H} \,\big|\, (\alpha, \Gamma) \overset{\tau}{\Rightarrow} \mathtt{H}}{\mathtt{I} \overset{\mathtt{t}}{\Rightarrow} \mathtt{I} \,\big|\, \mathtt{I} \overset{\mathtt{t}}{\Rightarrow} \theta \over \mathtt{e} \to \mathtt{t}}}\,[\equiv]}{\dfrac{\dfrac{(\alpha, (\kappa, \Gamma)) \overset{\tau}{\Rightarrow} \mathtt{H} \,\big|\, (\alpha, \Gamma) \overset{\tau}{\Rightarrow} \mathtt{H}}{\mathtt{I} \overset{\mathtt{t}}{\Rightarrow} \mathtt{I} \,\big|\, \mathtt{I} \overset{\mathtt{t}}{\Rightarrow} \theta \over \mathtt{e} \to \mathtt{t}}}{\chi \,\big|\, \chi}}\,[\mathbf{K}^2\text{-}\mathbf{U}]}$$

$$\dfrac{\text{a nurse} : \dfrac{\dfrac{\mathtt{o} \,\big|\, \mathtt{o}}{\rho \,\big|\, \rho} \over \mathtt{M} \overset{\sigma}{\Rightarrow} \mathtt{N} \,\big|\, \mathtt{M} \overset{\sigma}{\Rightarrow} \mathtt{N}}{\mathtt{e}} \qquad \text{did}_1 : \dfrac{\dfrac{(\alpha, (\kappa, \Gamma)) \overset{\tau}{\Rightarrow} \mathtt{H} \,\big|\, (\alpha, \Gamma) \overset{\tau}{\Rightarrow} \mathtt{H}}{\mathtt{I} \overset{\mathtt{t}}{\Rightarrow} \mathtt{I} \,\big|\, \mathtt{I} \overset{\mathtt{t}}{\Rightarrow} \theta} \over \mathtt{e} \to \mathtt{t}}}{\text{a doctor did}_1 : \dfrac{(\alpha, (\kappa, \Gamma)) \overset{\tau}{\Rightarrow} \mathtt{H} \,\big|\, (\alpha, \Gamma) \overset{\tau}{\Rightarrow} \mathtt{H}}{\dfrac{\mathtt{I} \overset{\mathtt{t}}{\Rightarrow} \mathtt{I} \,\big|\, \mathtt{I} \overset{\mathtt{t}}{\Rightarrow} \theta}{\mathtt{M} \overset{\sigma}{\Rightarrow} \mathtt{N} \,\big|\, \mathtt{M} \overset{\sigma}{\Rightarrow} \mathtt{N}} \over \mathtt{t}}}\,[\mathbf{K}^3\text{-}\backslash]$$

$$\dfrac{}{\text{a doctor did}_1 : (\alpha, (\kappa, \Gamma)) \overset{\mathtt{t}}{\Rightarrow} (\ldots, (\alpha, \Gamma))}\,[\mathbf{K}^3\text{-}\eta, \mathbf{K}^2\text{-}\mathbf{D}, \text{K-}\mathbf{D}, \mathbf{D}]$$

(100b) $[\![(100a)]\!] = \lambda(a, (Q, g)).\, Q\,(\lambda P.\lambda i.\, \{\langle P\, x, (\ldots, i)\rangle \mid \mathbf{doc}\, x\})\,(\ldots, (a, g))$

The output type of (99a) indicates that the result of evaluating this clause will contain a quantificational property. The input type of (100a) indicates that this clause will need to be evaluated in a context containing a quantificational property. Running the first clause and second clause in sequence, the output of the former satisfies the input of the latter, and the dependency is eliminated. The variable Q in (100b) is resolved to the lifted property $[\![\text{called everyone}]\!]$ defined in (99a).

(101a)

$$\dfrac{\dfrac{\dfrac{\vdots\ (99a)}{\text{a nurse (called eo)}^{\triangleright} : \Gamma \overset{\mathtt{t}}{\Rightarrow} (\ldots, (\kappa, \Gamma))}\,[\star]}{\text{a nurse (called eo)}^{\triangleright} : \dfrac{\Gamma \overset{\tau}{\Rightarrow} \mathtt{H} \,\big|\, (\ldots, (\kappa, \Gamma)) \overset{\tau}{\Rightarrow} \mathtt{H}}{\mathtt{t}}} \qquad \dfrac{\dfrac{}{\text{and} : \mathtt{t} \to \mathtt{t} \to \mathtt{t}}\,[\text{Lex}]}{\text{and} : \dfrac{\rho \,\big|\, \rho}{\mathtt{t} \to \mathtt{t} \to \mathtt{t}}}\,[\mathbf{U}] \qquad \dfrac{\dfrac{\vdots\ (100a)}{\text{a doc did}_1 : (\alpha, (\kappa, \Gamma)) \overset{\mathtt{t}}{\Rightarrow} (\ldots, (\alpha, \Gamma))}\,[\star]}{\text{a doc did}_1 : \dfrac{(\alpha, (\kappa, \Gamma)) \overset{\tau}{\Rightarrow} \mathtt{H} \,\big|\, (\ldots, (\alpha, \Gamma)) \overset{\tau}{\Rightarrow} \mathtt{H}}{\mathtt{t}}}$$

$$\dfrac{\text{a nurse called (called eo)}^{\triangleright} \text{ and a doc did}_1 : \dfrac{\Gamma \overset{\tau}{\Rightarrow} \mathtt{H} \,\big|\, (\ldots, (\alpha, \Gamma)) \overset{\tau}{\Rightarrow} \mathtt{H}}{\mathtt{t}}}{\text{a nurse (called eo)}^{\triangleright} \text{ and a doc did}_1 : \Gamma \overset{\mathtt{t}}{\Rightarrow} (\ldots, (\alpha, \Gamma))}\,[\text{K-}\eta, \mathbf{D}]\,[\text{K-/}, \text{K-}\backslash]$$

8 Discussion and comparison

The fragment outlined above draws on a great body of semantic research concerning scope and binding, more than we can or should talk about. In this section, we highlight and juxtapose a few important points of contact with the literature.

8.1 Hardt (1999) and other theories of ellipsis

Our semantics for sloppy ellipsis and paycheck anaphora is essentially that of Hardt 1999, but the fragment here improves upon Hardt's in several ways. First, we make no use of centering theory. Types aside, the dynamic semantics is rather vanilla. Discourse referents are pushed to the front of a context, but there is no privileged theoretical role to being in this position, and no referent is ever over-written by another. This is important because, as Sauerland (2007) points out, there can be more than one sloppy pronoun in a single ellipsis site:

(102) When a graduate student submits a paper, we expect them to have proofread it.
But when an undergraduate submits an assignment, we don't.

If each sloppily-retrieved antecedent had to be stored in a distinguished, destructively updated position, then one of the new referents in (102) would delete the other before its pronoun was able to retrieve it.[3]

Sauerland (2007) also points out that the antecedents of sloppy ellipsis sites may contain the traces of overtly extracted material, as in (103).

(103) What John eats depends on what Mary does.
But what John drinks doesn't.

If the ellipsis site in the first clause is identified with a simple pro-VP (so that it can be sloppily re-bound in the second clause), then it is unclear how it's supposed to bear the relevant syntactic relationship with the fronted relative pronoun 'what'. One solution to this is to follow Shan & Barker (2006) in treating the semantics of extraction gaps as yet another species of pronoun. Then, ellipsis to the body of a relative clause, as in (103) is no different than any other kind of sloppy ellipsis, semantically speaking. Of course the distribution of trace gaps should be quite carefully constrained to prevent, say, free or strict readings of traces, but plausibly this can be handled with a finer-grained type theory, one that distinguishes trace-dependencies from pronominal-dependencies (cf. our distinction in Section 3.1 between $\Rightarrow$ and $\rightarrow$, or Shan & Barker's distinction between $\mathsf{e} \setminus\!\!\setminus \alpha$ and $\mathsf{e} \triangleright \alpha$).

We might also add that where (103) shows that an expression can be overtly extracted from a Hardtian pro-VP, (104) shows that an expression can also be covertly scoped out of one.

(104) When Mary orders one of everything, John tries to as well.
But when Mary orders two of everything, he doesn't.

We needn't say anything new to capture this, though. The derivation of (104) follows straightforwardly from the derivations of (96) and (99). Again the only difference between (96) and the derivation of (104) will be in the types of the pronouns and the discourse referents. In this case, the ellipsis in the first clause represents anaphora to a quantificational property, as in (104). This entire 'tries to_0' meaning is then added to the context, creating a referent that is itself a dynamic request for a higher-order antecedent. When this referent is retrieved at doesn't_0, that request will be fulfilled by the local higher-order meaning of 'order two of everything'.

In a related vein, a significant challenge for anaphoric/semantic approaches to ellipsis such as Hardt's and ours has traditionally been the existence of case-matching effects in ellipsis, in which an elided XP assigns case to material "extracted out of" it exactly as if the XP was syntactically instantiated but unpronounced (e.g., Merchant 2001, Jäger 2005, Barker 2013). The type-transparency

3 We note that Hardt 1993, the precursor to Hardt 1999, did not make use of "center" indices, and did not have this problem.

of our semantics, in which the presence of unbound variable elements inside an expression is visible in the expression's type, allows us to account for these patterns. We begin by refining type e with a range of case-indexed subtypes $\mathsf{e}_{\mathsf{NOM}}$, $\mathsf{e}_{\mathsf{DAT}}$, etc. Then, for example, an accusatively typed gap of overt movement can be assigned type $\text{gap}_{0,\mathsf{ACC}} : (\mathsf{e}_{\mathsf{ACC}}, \Gamma) \overset{\mathsf{e}_{\mathsf{ACC}}}{\Rightarrow} \Gamma$, reflecting the fact that its antecedent/binder must also be accusatively typed, at least in languages with more robust case matching requirements than English.[4] It follows that a VP hosting an unbound extraction gap, e.g., a VP with type $(\mathsf{e}_{\mathsf{ACC}}, \Gamma) \overset{\mathsf{e}\to\mathsf{t}}{\Rightarrow} \Gamma$, will likewise announce the type of this expected filler, and similarly for any discourse referents generated by such VPs in order to anchor downstream ellipsis. If an elliptical pro-VP, like a gap, inherits the type of its antecedent, then a gapped pro-VP will constrain the type of its filler in exactly the same way as the antecedent gapped VP does, as desired.

Conversely, semantic theories of ellipsis are well suited to explaining various striking parallels between ellipsis and run-of-the-mill pronominal anaphora. As we have already seen in (98), VPs, like pronouns, can be sloppily bound. Like pronouns, elided VPs also give rise to covarying readings under focus-sensitive operators, donkey anaphora, and quantificational/modal subordination:

(105) I only JUMPED when you did
cf. Only JOHN did his homework

(106) When Oscar uses the copier or talks on the phone, I can't
cf. Whenever I see a linguist, I wave to 'em

(107) Everyone ran 20 miles or biked 100 miles, and everyone later wished they hadn't
cf. Every student wrote an article, and every student sent it to *L&P*

In the interest of space, we do not offer detailed analyses of these examples here (but see Charlow 2012). We simply note that a theory of ellipsis oriented around dynamic binding, such as our own, is well positioned to explain the parallels between ellipsis and pronominal anaphora, while traditional syntactic theories of ellipsis may face challenges explicating what the elided content in these examples is supposed to be, and on what basis it is allowed to go unpronounced.

In sum, type transparency coupled with a dynamic-anaphoric approach to ellipsis seems to occupy a sweet spot in the design space. The resulting theory inherits many of the strengths of syntactic and semantic theories of ellipsis, while avoiding common pitfalls of both.

8.2 Büring (2004) and Chierchia (2020)

Turning to crossover, the treatment here as a manifestation of the general asymmetries imposed by dynamic interpretation is in much the same spirit as recent work by Chierchia (2020). But the mechanisms that regulate referent introduction, and scope, and therefore crossover, are very different. Perhaps most importantly, Chierchia assumes a transformational theory of scope-taking and hypothesizes that scope never feeds binding. Discourse referents are introduced (for the most part) by thematic heads and lexical predicates. Crucially, they are not introduced by the semantics of any DP, or any type-shifted DP, or any abstraction operator responsible for creating the scope of any DP. Since pronouns are bound to discourse referents, and discourse referents are simply not introduced by scope-shifting mechanisms, there is no crossover.

4 In contrast, a regular old accusative pronoun would have type $\text{pro}_{0,\mathsf{ACC}} : (\mathsf{e}, \Gamma) \overset{\mathsf{e}_{\mathsf{ACC}}}{\Rightarrow} \Gamma$. It functions as an $\mathsf{e}_{\mathsf{ACC}}$, but doesn't impose any sorts of case restrictions on its binder. For details on how treating $\mathsf{e}_{\mathsf{ACC}}$ as a proper subtype of e allows this outcome, see the discussions of subtyping in Bernardi & Szabolcsi 2008, Charlow 2017.

As Chierchia notes, what actually prevents crossover in his fragment is the stratification of pronouns — which pick their referents out of the context — and traces — which are valued by a separate assignment function. The former are shifted by predicates, which push their arguments as referents onto the context; the latter is shifted by the object-language abstractions used to build scopes for quantifiers. This trick has precedents in at least Butler 2003 (building on Dekker 1994) and Büring 2004, who both also use it to prevent crossover.

The dynamic substrate of Chierchia's semantics is actually quite beside the point, as far as crossover is concerned, as witnessed by the fact that Büring's (2004) very similar analysis is static. The only thing the dynamics buys is the ability of objects to bind into adjuncts, as in (108).[5] If adjuncts like this are just conjuncts, and conjunction is dynamic, then the binding without c-command here is predicted.

(108) The dean interviewed every student in the presence of their advisor.

In particular, one of the things the dynamics *does not* buy Chierchia is garden-variety cases of donkey anaphora. This is because Chierchia, like Büring (2004), is *forced* to say that there is no binding out of DP. That is, none of the examples in (109) can constitute genuine cases of binding.

(109) Every student who wrote a paper submitted it.

(110) The dean of no college despises its students.

(111) Every student's advisor tries to help them.

The reason is that in each case, a quantifier would have to pass a discourse referent one way or another into its scope, but this is crucially forbidden. Instead, Chierchia, again like Büring, adopts an E-type approach to donkey anaphora. Chierchia's particular E-type flavor is a hybrid combining a traditional Cooper (1979)-style salient function with Elbourne (2005)-style NP ellipsis, as sketched in (112).

(112) [Every student who wrote a paper]$_i$ submitted $f(i)$(paper)

The NP ellipsis assumption is meant to guarantee that the pronoun has an actual linguistic antecedent, a "formal link" to the preceding discourse. But it also creates difficulties in exactly the kinds of sloppy configurations analyzed in the previous section. Indeed, all of the pronouns in (109) can be interpreted sloppily (Tomioka 1999), as in (113).

(113a) Every student who wrote a paper submitted it. Every student who wrote a squib did too.

(113b) The dean of no humanities division despises its students. But the dean of every sciences division does.

5 Chierchia argues that dynamic semantics *explains* the difference between traces and pronouns — they have different types, because they are valued by different parameters (the assignment function and the context, respectively), where Büring simply stipulates this difference. But this does not seem particularly fair. It's true that Büring evaluates all expressions relative to a single assignment, which values traces and pronouns alike, but that assignment is formally segregated into two entirely distinct sets of indices. It has, in effect, two domains, one for the variables attached to traces and the other for the variables attached to pronouns. This is clearly equivalent to having two distinct assignment parameters, one of which could be identified as the discourse context. Neither formalism explains by itself why traces and pronouns should run on different tracks, as it were, but both can claim to be implementations of whatever does.

Incidentally, the continuation semantics used here for scope-taking probably does have some explanatory purchase on distinguishing traces from pronouns, because there are no traces. Quantifiers "bind" their argument positions by being in them. It doesn't even make sense to ask why they don't bind pronouns through the same mechanism.

(113c) Every freshman's avidsor tries to help them. As does every senior's advisor.

To get the right interpretation of (113a), for example, the ellipsis site would have to contain 'submitted $f(i)$(squib)', but this VP does not occur in the antecedent. These sorts of examples motivate a genuinely *dynamic* account of donkey anaphora just as they do a dynamic account of ellipsis!

In any case, there is also good reason to think scope does in fact feed binding (Chierchia 2005, Barker 2012, Charlow 2019b), as the examples in (109) suggest. Chiefly, an embedded indefinite can antecede a (donkey / cross-clausal) pronoun *if and only if* the indefinite is not in the scope of any intervening (static) operators. For instance, consider the paradigm in (114).

(114a) Every time we chose to exclude a sample, we marked it with an X.

(114b) Every time we chose not to include a sample, we marked it with an X.

(114c) Every time we chose to exclude every sample, we marked it with an X.

The sentence in (114a) is an ordinary instance of donkey anaphora. According to Chierchia, its interpretation depends on the recovery of a salient function f from times to samples, say one that maps each time t to a sample we excluded at t. If there were times when we rejected multiple samples, then there will be more than one such function. In this case, Chierchia suggests that, as a default, we take it that all such functions ought to make the sentence true (that way we don't have to make an arbitrary choice).

The sentence in (114b) also has an ordinary donkey reading, on which it means exactly the same thing as (114a): namely, on those occasions when there was at least one sample that we chose not to include, we marked (all) such samples with an X. Crucially, here the indefinite takes inverse scope over the negation above it. There is no reading of (114b) that quantifies only over situations in which *no samples* were included.

Is this correlation between scope and binding predicted by the Cooper (1979) analysis of the pronoun? It's hard to see why it would be. Presumably one would have say that (114b) does not make any function from times to samples salient. But why would that be? Why not choose the *very same functions* as in (114a), the ones that map times t to samples excluded at t? There might be more than one, but that was the case in (114a) as well. Universally quantifying over them so as not to choose arbitrarily would render the sentence true if every time we excluded every sample, we marked every such excluded sample with an X. There's nothing weird about this reading; it just isn't something (114b) can mean.

For that matter, (114c) also seems problematic. It has no donkey reading at all. That pronoun in the scope must be free. The familiar dynamic explanation for this is that 'every sample', unlike 'a sample' is externally static, so no discourse referent is passed from the restrictor to the scope. But there's nothing that rules out ellipsis to the embedded NP; cf. (115).

(115) Every time we chose to exclude every sample, we made sure to mark Bill's with an X an everyone else's with a Y

Then it should just be a matter of finding some salient functions from times to samples. Again, functions from times t to samples excluded at t seem as reasonable here as in the previous cases, and again, universally quantifying over these would yield the truth conditions that whenever all samples were excluded, all excluded samples were marked with an X.

A few words on the argument here. First, there's nothing special about negation per se (*pace* Elliott 2020). The same paradigm could be constructed with any operator that forces the indefinite to take inverse scope in order to bind a pronoun. Second, what we've pointed out in connection to (114) is a potential *overgeneration* problem for the static account of donkey anaphora, but the claim we want to support is that the broader prohibition on binding from scope positions in fact creates an *undergeneration* problem. The logic is this: the only reason that successful donkey binding is correlated with the indefinite taking widest scope in the restrictor is that this is the only position from which its discourse referents will be accessible to the pronoun. But this presupposes that it creates discourse referents where it scopes.

Finally, the particular assumptions that Chierchia makes about the syntax-semantics interface (which are standard), coupled with his hypothesis that referents are only created in thematic positions (which is non-standard), conspire to rule out any obvious derivation for sentences like (116). This would seem to be a problem for Büring (2004) as well.

(116) A referee rejected every paper she reviewed.

The sentence should have a surface-scope reading, true if some referee exercised a blanket policy of rejection for her assignments. The problem is that the universal quantifier, being in object position, must scope over the subject to be interpreted, and then the subject must scope over the universal in order to get the truth conditions right (the harsh referee does not vary with papers). But in moving over the subject position, the universal will carry the pronoun in its restrictor along with it. Ordinarily, the raised subject could still bind that pronoun, since it outscopes it, but it would have to do so from its scope position, as the discourse referent introduced by the subject trace is too low to bind the raised object NP. Presumably some sort of reconstruction could be appealed to, but it is notable that such appeal is necessary.

In contrast, the analysis we've presented here is dynamic through and through, allows scope to feed binding, and therefore binding out of DP in both its donkey and inverse-linking varieties.

8.3 Barker & Shan (2008)

Our analysis incorporates what we take to be the fundamental insight about crossover from Shan & Barker (2006) and Barker & Shan (2008): that when semantic types track scope inversions, the grammar can prevent inverse binding parametrically. Like the previous authors, we use continuations to stratify effects into levels. Discourse referents flow from left-to-right *per level*, and never from top-to-bottom. Since scope inversion would require a top-to-bottom transmission of referents, it is impossible. Also like the previous authors, we guarantee that binding is a polymorphic and type-transparent process. Any kind of value can be pushed for downstream anaphora, and the types of expressions containing pushed referents, as well as the types of expressions containing unbound pronouns, differ from those of pure expressions. Moreover, referents in both fragments constitute a linear resource; they are consumed in the course of binding.

But unlike Barker & Shan (2008), we assume a dynamic semantics in the traditional sense that donkey indefinites do not outscope their donkey pronouns. Instead, they modify contexts, and those context updates will either thread their way through a discourse, valuing pronouns as they go, or will be captured and quantified over by various operators. This avoids a number of overgeneration issues that threaten Barker & Shan's split-scope analysis, detailed in Charlow 2010 and Barker & Shan 2014: Ch. 10. It also permits a simple characterization of what happens at scope island boundaries: all continuation arguments must be saturated, so that towers are fully lowered. This

guarantees that every quantifier will have combined with a scope, which prevents any further scope-taking. But it does *not* prevent information from flowing through the state, which is what indefinites and pronouns use to communicate. See Charlow 2014 for discussion.

8.4 Dynamic Semantics

Architecturally speaking, the fragment here is extremely close to that of Charlow 2014. The difference is in the types and the system of pronouns that give rise to those types. Charlow models contexts as flat lists of values. Pronouns use integers as indices to target particular positions in the list. As in many other sequence-based dynamic theories (e.g., Vermeulen 1993, Dekker 1994, van Eijck 2001), the relationship between pronouns and their antecedents is variable-free, in the sense that binders and bindees share no formal or syntactic connection. That is, even though there is indexation on the pronouns, there is no *co*-indexation between the pronouns and their antecedents. As Dekker (2012) puts it, the indices are fundamentally indexical.

But the sequence-encoding of contexts does not fit very cleanly into simple type systems. Most dynamic frameworks simply restrict contexts to a single type (e), which guarantees well-typedness (presuming indices on pronouns never outrun the lengths of contexts), but is clearly not up to the tasks of Section 7. Allowing heterogeneous lists, as in Charlow 2014, opens the semantic door to all of the phenomena that we've derived, but at some cost in type safety or type complexity (see the discussion in Section 2). One contribution of this paper, then, is to recover those static type assurances while still permitting higher-order anaphora (see also Charlow 2019a for another way to go). In addition, having an articulated type structure for contexts provides exactly the information needed to prevent crossover, since the grammar can now see the difference between a pure expression and an expression with an anaphoric dependency.

But we should also point out that these two contributions are to some extent dissociable. Obviously one could choose to adopt the parametrized monadic dynamic semantic framework with polymorphic, statically-typed, heterogeneous contexts as we've laid it out, but seek a different explanation for crossover phenomena. But one could also adopt something very close to our analysis of crossover without going all in on the monadic semantics. The fundamental mechanism by which crossover is prevented is that when expressions are lowered, the inner computations (which are in the process of being scoped over) are executed in an empty context. That's really it.

Say, for instance, you wanted a more traditional view of context change, so that dynamic meanings are simply relations on finite maps (partial assignments) from variables to entities. You could certainly in principle combine this with a continuation account of scope inversion (though, without monads, anything with a dynamic effect will have to be a quantifier, so islands will get sticky). Then all you have to do to prevent crossover is to make sure that lower-levels of towers are run with empty maps. Let $\mathsf{T} \equiv \mathsf{s} \to \mathsf{s} \to \mathsf{t}$ stand for the type of relations on variable maps s. And say we have the following lexical entries.

(117a) called : $\mathsf{e} \to \mathsf{e} \to \mathsf{T}$
$[\![\text{called}]\!] = \lambda d \lambda e \lambda g \lambda h.\, g = h \wedge \mathbf{call}\, d\, e$

(117b) pro_u's mom : $(\mathsf{e} \to \mathsf{T}) \to \mathsf{T}$
$[\![\text{pro}_u\text{'s mom}]\!] = \lambda P \lambda g \lambda h.\, P\, g_u\, g\, h$

(117c) everyone^u : $(\mathsf{e} \to \mathsf{T}) \to \mathsf{T}$
$[\![\text{everyone}^u]\!] = \lambda Q \lambda g \lambda h.\, g = h \wedge \forall d.\, Q\, d\, g^{u \mapsto d}\, h$

Then, we can keep all the continuation rules exactly as above (the ones that start with [K]), and fix a version of [**D**] for the contexts-as-maps denotations.

(118) $$\left[\!\!\left[\dfrac{E : \dfrac{\rho \mid \mathsf{T}}{\mathsf{T}}}{E : \rho}\,[\mathbf{D}] \right]\!\!\right] \coloneqq [\![E]\!]\,(\lambda m.\lambda g.\lambda h.\,\exists j.\, m\,\emptyset\, j \wedge h = g \cup j)$$

As desired, this rule runs m, the computation that is being scoped over, in an empty context $\emptyset$. Whatever output contexts j emerge from that computation are merged with the current input g, and life continues. This ensures that crossed-over pronouns, as in (119) will not pick up the discourse referents introduced by subsequent expressions.

(119)
$$\dfrac{\dfrac{\dfrac{\text{pro}_u\text{'s mom} : \dfrac{\mathsf{T} \mid \mathsf{T}}{\mathsf{e}}\,[\mathbf{U}] \qquad \text{called everyone}^u : \dfrac{\mathsf{T} \mid \mathsf{T}}{\mathsf{e} \to \mathsf{T}}\,[\text{K-}\mathbf{U}]}{\text{pro}_u\text{'s mom} : \dfrac{\dfrac{\rho \mid \rho}{\mathsf{T} \mid \mathsf{T}}}{\mathsf{e}} \qquad \text{called everyone}^u : \dfrac{\dfrac{\mathsf{T} \mid \mathsf{T}}{\rho' \mid \rho'}}{\mathsf{e} \to \mathsf{T}}}\,[\text{K}^2\text{-}\backslash]}{\text{pro}_u\text{'s mom called everyone}^u : \dfrac{\dfrac{\mathsf{T} \mid \mathsf{T}}{\mathsf{T} \mid \mathsf{T}}}{\mathsf{T}}}\,[\text{K-}\mathbf{D}]}{\text{pro}_u\text{'s mom called everyone}^u : \dfrac{\mathsf{T} \mid \mathsf{T}}{\mathsf{T}}}\,[\mathbf{D}]}{\text{pro}_u\text{'s mom called everyone}^u : \mathsf{T}}$$

Of course, what we lose in this presentation is type-safety! Anaphorically active denotations have type T. So do pure, pronoun-free denotations. There's no way for the grammar to know whether the lower-level meaning can really compute what it needs to compute in the context of an empty map. So well-typed programs may crash. Indeed, crossover configurations like the one in (119) certainly will. We take no stand on whether grammatical-but-meaningless is a defensible way to think about crossover violations. The point here is just that something in the semantic spirit of the Shan & Barker (2006) story about crossover can be replicated in a variety of systems.

9 Conclusion

In this paper we've tried to do two main things. First, we've *parameterized* the monadic dynamic semantics of Charlow 2014, so that input and output context types can differ. Taking advantage of this, we defined a pair of combinators z and s that generate typed *lenses* for reading and modifying contexts of various types, and we have identified these lenses as the meanings of pronominal indices. Together with the parameterized state, such lenses ensure that the types of expressions reveal exactly what sorts of dynamic effects they execute. This allows for well-typed cross-categorial dynamic effects, including the higher-order anaphora seen in sloppy binding phenomena, without threat of the semantics becoming paradoxical. Second, we've exploited the newfound type transparency to supplement this dynamic grammar with ideas from variable-free semantics concerning crossover,

in effect generalizing Shan & Barker's (2006) treatment of crossover and superiority to a dynamic setting. Thus we combine a very successful and well-studied approach to cross-clausal and donkey binding with a very successful approach to scope and evaluation-order asymmetries, all within a directly compositional framework with a simple theory of types.

References

Atkey, Robert. 2009. Parameterised notions of computation. *Journal of Functional Programming* 19(3-4). 335–376. https://doi.org/10.1017/S095679680900728X.

Barker, Chris. 2012. Quantificational binding does not require c-command. *Linguistic Inquiry* 43(4). 614–633. https://doi.org/doi:10.1162/ling_a_00108.

Barker, Chris. 2013. Scopability and sluicing. *Linguistics and Philosophy* 36(3). 187–223. https://doi.org/10.1007/s10988-013-9137-1.

Barker, Chris & Chung-chieh Shan. 2008. Donkey anaphora is in-scope binding. *Semantics and Pragmatics* 1(1). 1–46. https://doi.org/10.3765/sp.1.1.

Barker, Chris & Chung-chieh Shan. 2014. *Continuations and natural language*. Oxford: Oxford University Press. https://doi.org/10.1093/acprof:oso/9780199575015.001.0001.

Bernardi, Raffaella & Anna Szabolcsi. 2008. Optionality, scope, and licensing: An application of partially ordered categories. *Journal of Logic, Language and Information* 17(3). 237–283. https://doi.org/10.1007/s10849-008-9060-y.

Bittner, Maria. 2001. Surface composition as bridging. *Journal of Semantics* 18(2). 127–177. https://doi.org/10.1093/jos/18.2.127.

Bott, Oliver & Wolfgang Sternefeld. 2017. An event semantics with continuations for incremental interpretation. *Journal of Semantics* 34(2). 201–236.

Brasoveanu, Adrian & Jakub Dotlačil. 2020. *Computational cognitive modeling and linguistic theory*. Springer Nature.

Büring, Daniel. 2004. Crossover situations. *Natural Language Semantics* 12(1). 23–62. https://doi.org/10.1023/B:NALS.0000011144.81075.a8.

Butler, Alastair. 2003. Predicate logic with barriers and its locality effects. In *Proceedings of Sinn und Bedeutung*, vol. 7, 70–80.

Carette, Jacques, Oleg Kiselyov & Chung-chieh Shan. 2009. Finally tagless, partially evaluated: tagless staged interpreters for simpler typed languages. *Journal of Functional Programming* 19(5). 509–543. https://doi.org/10.1017/S0956796809007205.

Charlow, Simon. 2010. Two kinds of binding out of DP. 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. https://doi.org/10.1007/978-3-642-31482-7_27.

Charlow, Simon. 2014. *On the semantics of exceptional scope*. New York University Ph.D. thesis. https://semanticsarchive.net/Archive/2JmMWRjY/.

Charlow, Simon. 2017. Post-suppositions and semantic theory. Accepted at *Journal of Semantics*. https://ling.auf.net/lingbuzz/003243.

Charlow, Simon. 2019a. A modular theory of pronouns and binding. Unpublished ms., Rutgers University. https://ling.auf.net/lingbuzz/003720.

Charlow, Simon. 2019b. Static and dynamic exceptional scope. Accepted at *Journal of Semantics*. https://ling.auf.net/lingbuzz/004650.

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.

Chierchia, Gennaro. 2020. Origins of weak crossover: when dynamic semantics meets event semantics. *Natural Language Semantics* 28(1). 23–76. https://doi.org/10.1007/s11050-019-09158-3.

Cooper, Robin. 1979. The interpretation of pronouns. In Frank Heny & Helmut S. Schnelle (eds.), *Syntax and semantics, volume 10: Selections from the Third Groningen Round Table*, 61–92. New York: Academic Press.

Danvy, Olivier. 1998. Functional unparsing. *Journal of Functional Programming* 8(6). 621–625. https://doi.org/10.1017/S0956796898003104.

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. https://doi.org/10.3765/salt.v4i0.2459.

Dekker, Paul. 2012. *Dynamic semantics* (Studies in Linguistics and Philosophy 91). Dordrecht: Springer Science+Business Media. https://doi.org/10.1007/978-94-007-4869-9.

van Eijck, Jan. 2001. Incremental dynamics. *Journal of Logic, Language and Information* 10(3). 319–351. https://doi.org/10.1023/A:1011251627260.

Elbourne, Paul. 2005. *Situations and individuals*. Cambridge, MA: MIT Press.

Elliott, Patrick. 2020. Crossover and accessibility in dynamic semantics. Unpublished ms.

Fridlender, Daniel & Mia Indrika. 2000. Do we need dependent types? *Journal of Functional Programming* 10(4). 409–415. https://doi.org/10.1017/S0956796800003658.

Groenendijk, Jeroen & Martin Stokhof. 1991. Dynamic predicate logic. *Linguistics and Philosophy* 14(1). 39–100. https://doi.org/10.1007/BF00628304.

Hardt, Daniel. 1993. *VP ellipsis: Form, meaning, and processing*. University of Pennsylvania Ph.D. thesis. https://repository.upenn.edu/dissertations/AAI9331786.

Hardt, Daniel. 1999. Dynamic interpretation of verb phrase ellipsis. *Linguistics and Philosophy* 22(2). 187–221. https://doi.org/10.1023/A:1005427813846.

Heim, Irene. 1982. *The semantics of definite and indefinite noun phrases*. University of Massachusetts, Amherst Ph.D. thesis. https://semanticsarchive.net/Archive/Tk0ZmYyY/.

Heim, Irene & Angelika Kratzer. 1998. *Semantics in generative grammar*. Oxford: Blackwell.

Jacobson, Pauline. 1999. Towards a variable-free semantics. *Linguistics and Philosophy* 22(2). 117–184. https://doi.org/10.1023/A:1005464228727.

Jacobson, Pauline. 2014. *Compositional semantics: An introduction to the syntax/semantics interface*. Oxford: Oxford University Press.

Jäger, Gerhard. 2005. *Anaphora and type logical grammar*. Dordrecht: Springer. https://doi.org/10.1007/1-4020-3905-0.

Kamp, Hans. 1981. A theory of truth and semantic interpretation. In Jeroen Groenendijk, Theo M. V. Janssen & Martin Stokhof (eds.), *Formal Methods in the Study of Language*, 277–322. Mathematical Centre Amsterdam.

Kiselyov, Oleg. 2012. Typed Tagless Final Interpreters. In Jeremy Gibbons (ed.), *Generic and Indexed Programming: International Spring School, SSGIP 2010, Oxford, UK, March 22-26, 2010, Revised Lectures*, 130–174. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-32202-0_3.

Martin, Scott. 2016. Supplemental update. *Semantics and Pragmatics* 9(5). 1–61. https://doi.org/10.3765/sp.9.5.

Merchant, Jason. 2001. *The syntax of silence: Sluicing, islands, and the theory of ellipsis*. Oxford: Oxford University Press.

Muskens, Reinhard. 1995. Tense and the logic of change. In Urs Egli, Peter E. Pause, Christoph Schwarze, Arnim von Stechow & Götz Wienold (eds.), *Lexical Knowledge in the Organization of Language*, 147–183. Amsterdam: John Benjamins. https://doi.org/10.1075/cilt.114.08mus.

Muskens, Reinhard. 1996. Combining Montague semantics and discourse representation. *Linguistics and Philosophy* 19(2). 143–186. https://doi.org/10.1007/BF00635836.

Pierce, Benjamin C. 2002. *Types and programming languages*. Cambridge, MA: MIT Press.

Rett, Jessica. 2022. A typology of semantic entities. In Daniel Altshuler (ed.), *Linguistics meets philosophy*, chap. 9. Oxford: Cambridge University Press.

Rooth, Mats. 1985. *Association with focus*. University of Massachusetts, Amherst Ph.D. thesis.

Sauerland, Uli. 2007. Copying vs. structure sharing: a semantic argument. *Linguistic Variation Yearbook* 7(1). 27–51. https://doi.org/10.1075/livy.7.03sau.

Schwarz, Bernhard. 2000. *Topics in ellipsis*. University of Massachusetts, Amherst Ph.D. thesis. https://scholarworks.umass.edu/dissertations/AAI9960789.

Shan, Chung-chieh & Chris Barker. 2006. Explaining crossover and superiority as left-to-right evaluation. *Linguistics and Philosophy* 29(1). 91–134. https://doi.org/10.1007/s10988-005-6580-7.

Szabolcsi, Anna. 1989. Bound variables in syntax (are there any?) In Renate Bartsch, Johan van Benthem & Peter van Emde Boas (eds.), *Semantics and contextual expressions*, 295–318. Dordrecht: Foris.

Tomioka, Satoshi. 1999. A sloppy identity puzzle. *Natural Language Semantics* 7(2). 217–241. https://doi.org/10.1023/A:1008309217917.

Vermeulen, C. F. M. 1993. Sequence semantics for dynamic predicate logic. *Journal of Logic, Language and Information* 2(3). 217–254. https://doi.org/10.1007/BF01050788.

Wadler, Philip. 1994. Monads and composable continuations. *LISP and Symbolic Computation* 7(1). 39–56. https://doi.org/10.1007/BF01019944.