Title: Unifying the E-type and plural dynamic approaches to improper scope phenomena

Authors: Steven Abney, Ezra Keshet
Orcid IDs: 0000-0002-7467-6690, 0000-0002-9359-2062

Affiliation: University of Michigan, Ann Arbor, Michigan, U.S.A.

Abstract

There is a variety of constructions in which singular and plural pronouns are not syntactically bound by their antecedents, such as donkey anaphora (Geach 1962), paycheck pronouns (Karttunen 1969), quantificational subordination (Sells 1985), and even simple cross-sentential anaphora. Many formal systems have been proposed to capture them, with ever-increasing complexity. This paper accounts for the same range of phenomena with a radically simpler system that we call **FOL-PLUS.** It extends First Order Logic (FOL) with three innovations: (i) bound variables are marked in-place with brackets, rather than appearing on their binder, e.g., $\ulcorner\exists\ldots[x]\ldots\urcorner$ instead of $\ulcorner\exists x\ldots x\ldots\urcorner$, (ii) a summation operator $\Sigma_x\phi$ combines all individuals x satisfying ϕ, and (iii) subformulas may be stored and retrieved via new, capital-letter formula labels. Unlike **E-type** approaches (Evans 1977), which pair pronouns with intuitively "salient" descriptions, FOL-PLUS formalizes the connection between all pronouns and their antecedents without requiring more and more complex, nested situation structures (Heim 1990; Elbourne 2005). And unlike **dynamic** approaches (Kamp 1981; Heim 1983; Groenendijk and Stokhof 1991; van den Berg 1996; Brasoveanu 2007), FOL-PLUS is completely static and relies only on a single variable assignment (as contrasted, for example, with the nondeterministic sets of sets of assignments required in van den Berg (1996)). The paper compares FOL-PLUS directly to these two approaches: a Heim and Kratzer (1998) style compositional translation system from natural language phrase structures into FOL-PLUS is given, allowing a direct comparison with similar E-type systems; and a formal proof is provided that FOL-PLUS has the same expressive power as existing dynamic systems. Finally, the paper introduces the notion of a **discourse block**, the somewhat surprising, single locus for all three FOL-PLUS innovations: the scope of bound variables, the location of summation operators, and the storage and retrieval of subformulas.

Keywords: E-type anaphora, dynamic semantics, plural dynamic logic, dynamic predicate logic, improper scope, relational algebra

Declarations

Authors are listed in alphabetic order. Both authors contributed equally to all aspects of the research and preparation of the work. The authors have no relevant financial or non-financial interests to disclose.

Acknowledgements

We are grateful to Richmond Thomason for detailed comments. Remaining errors are of course our own.

1 Introduction

1.1 Problem setting

Much research on anaphora in natural language centers around the following cluster of phenomena, in which a pronoun has an antecedent that does not syntactically bind it:

(1) a. *Cross-sentential anaphora*
A dog sauntered in. It sat down and barked.

b. *Summation pronouns*
Most students$_1$ wrote a paper$_2$. They$_1$ left them$_2$ on my desk.

c. *Paycheck Pronouns* (Karttunen 1969; Jacobson 2000)
The employee who saved her paycheck was wiser than the one who cashed it.

d. *Donkey pronouns* (Geach 1962)
Every farmer who owns a donkey beats it.

e. *Quantificational Subordination* (Karttunen 1969; Sells 1985)
Most students wrote a paper. Some of them even turned it in.

If the antecedents of these pronouns are treated as quantifiers—as they usually are—these examples are problematic because the pronouns lie outside the conventional scope of the quantifiers and thus should not be able to take the quantifiers as antecedents. For this reason, we adopt the term **improper scope phenomena** for the general class.[1]

There are two major approaches to improper scope phenomena. One, often called the **E-type** approach, follows Evans (1977) in assuming a class of E-type pronouns akin to definite descriptions, picking out a unique referent satisfying some description. For instance, the pronouns in (1) might denote *the dog who sauntered in*, *the donkey the farmer owns*, *the students who wrote a paper*, etc. This approach is most often employed by semanticists working in a static conception of semantics, for instance as proposed by Montague (1970). The main weakness of the approach is the difficulty of defining precisely what the "salient descriptions" are by which E-type pronouns identify a unique referent.

The second approach to improper scope phenomena developed within the tradition of **dynamic semantics**. Under this approach, a context state is maintained through the discourse, in which antecedents store and pronouns retrieve discourse referents. For instance, one may take an indefinite DP like *a dog* in (1a) to store an individual (namely, a particular dog) in the context state as the value for variable x. A later pronoun, even one in a different sentence, can retrieve the value of x from the context state. Unlike most E-type systems, dynamic accounts fully formalize the relationship between antecedents and pronouns. And yet, in attempting to capture the full range of phenomena

[1]The term *E-type anaphora* is frequently used, after Evans (1977, 1980), but we reserve *E-type* for analyses stemming from the one that Evans proposes, not the empirical phenomena.

mentioned above, dynamic semanticists have found it necessary to propose ever more complex context states. Moreover, dynamic systems struggle with cases that the E-type approach captures easily, such as paycheck pronouns.

1.2 Back to basics

We present here a new logic for improper scope phenomena that builds directly on First-Order predicate Logic (FOL) with a plural domain. To this classical foundation, we make only three alterations: a form of unselective binding where existentially bound variables are marked in-place instead of on their binder, a summation operator to collect multiple values for a variable, and labels to store and retrieve formulas. The resulting logic looks like (2):

(2) a. A girl saw a dog $\rightsquigarrow \exists(\textsc{girl}([x]) \wedge \textsc{dog}([y]) \wedge \textsc{saw}(x, y))$
b. The girls who saw a dog $\rightsquigarrow \Sigma_x(\textsc{girl}([x]) \wedge \textsc{dog}([y]) \wedge \textsc{saw}(x, y))$
c. Every girl who saw a dog pet it
$\rightsquigarrow \Sigma_x({}^A(\textsc{girl}([x]) \wedge \textsc{dog}([y]) \wedge \textsc{saw}(x, y))) \;=\; \Sigma_x({}_A(\textsc{pet}(x, y)))$

Simply put, the $\ulcorner\exists\urcorner$ and $\ulcorner\Sigma_x\urcorner$ operators existentially bind all bracketed variables in their scope, and $\ulcorner\Sigma_x\phi\urcorner$ denotes all values of x that satisfy ϕ. Superscript labels like $\ulcorner{}^A\phi\urcorner$ store ϕ for later retrieval via a subscript label: $\ulcorner{}_A\psi\urcorner$. And that is the absolute entirety of the system.

We call the resulting system FOL-PLUS (FOL with Plurals, Labels, Unselective binding, and Summation). It operates over the single, standard variable assignment of FOL, rather than the (sets of) sets of assignments employed by dynamic systems. And as a static system, it avoids the non-determinism which further complicates dynamic logics. The only new operator in FOL-PLUS is $\ulcorner\Sigma\urcorner$, unlike the many special operators assumed, e.g., by Brasoveanu (2007). In this way, FOL-PLUS is quite close to the static systems employed by E-type theorists, just with a formal mechanism for deriving improper binding, akin to dynamic semantics. Finally, despite its bare-bones simplicity, FOL-PLUS derives all the empirical results of both the dynamic and E-type approaches.

We take this to be the main contribution of our paper; namely, showing how such a basic system can still derive the results of its more complex predecessors. In a sense, this also reveals something of the fundamental structure of natural-language discourse, the essential elements any approach to improper scope must include: some long-distance connection between indefinites and their binders; some method of summing, or quantifying, over variables; and some higher form of anaphora, whether formula labels in FOL-PLUS, the salient predicates of E-type approaches, or the state complexification of dynamic systems.

Beyond simply describing these three essential elements, though, our research has also revealed a somewhat surprising connection between them, which we describe via the concept of a **discourse block**. It is not a new idea that discourse is organized into blocks comprising one or more natural language clauses. Our new observation is that three seemingly unrelated phenomena intersect at the block level:

The scope of indefinites Discourse blocks mark the scope of all indefinite expressions within them (excepting those within another, embedded block).

The scope of sentential operators Discourse blocks are introduced by sentential operators: existential closure, including existential closure below negation, and our new summation operator Σ. While it may be unsurprising that indefinites take scope below existential closure, it is less obvious that they must scope below an operator summing up individual denotations.

The locus of discourse anaphora Finally, and perhaps most surprisingly, only full discourse blocks may serve as antecedents to a form of anaphora that stores and retrieves full formulas. Furthermore, only full discourse blocks themselves may be anaphoric to previous blocks.

We have taken pains in this paper to facilitate easy comparison of FOL-PLUS to previous approaches. To that end, the remainder of this section presents the formal definition of FOL-PLUS, comparable to the logical meta-languages usually presented in the dynamic logic literature. Section 2 presents a translation system from natural language into this meta-language, based on the familiar system in Heim and Kratzer (1998), also commonly used in E-type approaches. In so doing, we also give the empirical results of our approach. Section 3 compares FOL-PLUS to Dynamic Predicate Logic, showing (contra Groenendijk and Stokhof 1991) that a static system can, in fact, achieve the same expressive power as their seminal dynamic system. Finally, section 4 discusses the two main previous approaches to improper scope phenomena.

1.3 Formal definition of FOL-PLUS

Without further ado, we introduce our logic, First Order Logic with Plurals, Labels, Unselective binding, and Summation (FOL-PLUS). It takes as its starting point standard first order predicate logic (FOL).

1.3.1 FOL

The interpretation function $[\![\phi]\!]$ is defined with respect to a domain $\mathcal{D}$ of individuals and a model $\mathcal{M}$ over $\mathcal{D}$, but the model is fixed for a given discourse and thus we do not include it as an explicit parameter of $[\![\phi]\!]$. Where g and h are assignment functions from variables to $\mathcal{D}$, we define:

(3)
a. $g\backslash V \quad \triangleq \{\langle v,x\rangle \in g : v \notin V\}$ [domain subtraction]
b. $g[V]h \quad$ iff $g\backslash V = h\backslash V$ [partial equivalence]
c. $C_{x,y,\ldots}\phi(g,h) \quad$ iff $g[x,y,\ldots]h$ and $[\![\phi]\!]^h = 1$ [variable closure]

In (3c), note that $C_{x,y,\ldots}\phi$ denotes a relation that is applied to arguments g and h. The FOL semantic-value function is defined in Figure 1.

In FOL, atomic formulas '$\text{P}(x,y,\ldots)$' are true if the values for variables $x,y,\ldots$ provided by the assignment g satisfy the predicate P in the model $\mathcal{M}$.

(4) Term denotations are individuals

a. $[\![x]\!]^g = g(x)$ [simple terms]

(5) Formula denotations are 1 or 0

a. $[\![\textsc{p}(\tau_1, \tau_2, \ldots)]\!]^g = 1$ iff $\langle [\![\tau_1]\!]^g, [\![\tau_2]\!]^g, \ldots \rangle \in \mathcal{M}(\textsc{p})$ [predication]

b. $[\![\tau_1 = \tau_2]\!]^g = 1$ iff $[\![\tau_1]\!]^g = [\![\tau_2]\!]^g$ [equality]

c. $[\![\neg\phi]\!]^g = 1$ iff $[\![\phi]\!]^g = 0$ [negation]

d. $[\![\phi \wedge \psi]\!]^g = 1$ iff $[\![\phi]\!]^g = [\![\psi]\!]^g = 1$ [conjunction]

e. $[\![\exists x, y, \ldots \phi]\!]^g = 1$ iff $\exists h \,.\, C_{x,y,\ldots}\phi(g, h)$ [existential quant.]

Figure 1: Formal Specification of FOL

A negated formula is true whenever its component formula is false. Conjunctions require both component formulas to be true. Finally, existential quantification over variables $x, y, \ldots$ is true whenever there exist values for $x, y, \ldots$ that make the embedded formula true, as is more easily seen after substituting the definition of $C_{x,y,\ldots}\phi$ into (5e):

(6) $[\![\exists x, y, \ldots \phi]\!]^g = 1$ iff $\exists h \,.\, g[x, y, \ldots]h \wedge [\![\phi]\!]^h{=}1$

Our definition is nonconventional only in that we have introduced $C_{x,y,\ldots}\phi$ as a name for the relation between the "outer" contextual assignment g and the "inner" contextual assignment h that arise in (6). The motivation is as follows. Although we treat $\exists x, y, \ldots$ syncategorematically in (5e), one can think of it as a sentential operator. In FOL, the semantic value of a sentence, $[\![\phi]\!]$, can be thought of as a function from an assignment to a truth value; thus a sentential operator takes a function from assignment to truth value and provides a function from assignment to truth value, which is to say, a sentential operator is a 2-place relation over assignments. $C_{x,y,\ldots}$ represents a generic sentential operator that we use to implicitly define $\exists x, y, \ldots$.

In particular, $C_{x,y,\ldots}$ is a general *closure operator.* Even if a formula ϕ is sensitive to the values of $x, y, \ldots$, the formula $C_{x,y,\ldots}\phi$ is not, and in that sense $C_{x,y,\ldots}$ closes the variables $x, y, \ldots$. Formally:

(7) a. (The value of) an expression ϕ is **sensitive to** (the value of) variable x just in case there are assignments h and h' such that $h[x]h'$ and $[\![\phi]\!]^h \neq [\![\phi]\!]^{h'}$.

b. An operator ω **closes** a variable x iff, for all formulas ϕ (including those that are sensitive to x), the formula $\omega\phi$ is insensitive to x.

Under this definition, $C_{x,y,\ldots}$ and hence $\exists x, y, \ldots$, close the variables $x, y, \ldots$.

Closure operators will play a special role in what follows. We consider each closure operator to introduce a new **block,** and in FOL-PLUS, the block delimits the scope of variables. To be precise:

(8) a. A **closure expression** is an expression $C\phi$ in which C is a closure

operator.

b. A **closed block** consists of all material contained in a closure expression, excluding any material contained in more deeply-embedded closure expressions. (Note that the x subscript in $C_x\phi$ lies within the block $C_x\phi$.)

c. The **root block** consists of any material not contained in an embedded closed block.

d. The **blocks** are the closed blocks and the root block.

1.3.2 FOL-PLUS

As a plural logic, the domain $\mathcal{D}$ of FOL-PLUS consists of pluralities, which for technical simplicity we take to be sets of **atoms,** including singleton and empty sets. In addition, as mentioned above, FOL-PLUS extends FOL in the following three ways: (i) variables to be closed are marked in-place using brackets: $\ulcorner[x]\urcorner$, and these are said to be *local variables* of their containing block; (ii) new terms $\ulcorner\Sigma_x\phi\urcorner$ denote the union of all values for the variable x that satisfy the formula ϕ; and (iii) subformulas may be labeled with capital letters and then retrieved later. We first present the FOL-PLUS system in its entirety, and then we will discuss these three extensions in more detail.

Syntax. A countably infinite set of variables $x, y, \ldots$ is provided, along with a disjoint, countably infinite set of **formula labels** $X, Y, \ldots$. A *term* is a variable, a variable enclosed in brackets $\ulcorner[x]\urcorner$, or an expression $\ulcorner\Sigma_x\zeta\urcorner$ where x is a variable and ζ is a formula or labeled formula. A *formula* is an expression whose form is one of: $\ulcorner\mathrm{P}(\tau_1, \ldots, \tau_n)\urcorner$, $\ulcorner\neg\phi\urcorner$, $\ulcorner\phi \wedge \psi\urcorner$, or $\ulcorner\exists\zeta\urcorner$, where P is an n-place predicate symbol, τ_i are terms, ϕ and ψ are formulas, and ζ is either a formula or labeled formula. A *half-labeled formula* is an expression $\ulcorner{}_X\phi\urcorner$ where X is a formula label and ϕ is a formula, and a *labeled formula* is an expression $\ulcorner{}^X\eta\urcorner$ where X is a formula label and η is a formula or half-labeled formula. (Intuitively, a formula may have at most one superscript label and one subscript label, in that order, and may only be labeled if it is the argument of a closure operator.)

Semantics. The domain $\mathcal{D}$ is the power set of Ω, the set of atoms. A model $\mathcal{M}$ over $\mathcal{D}$ and a label-value assignment $\mathcal{A}$ are assumed to be fixed for a given discourse. The domain of $\mathcal{A}$ is the set of formula labels, and $\mathcal{A}$ assigns a formula of FOL-PLUS to each. An expression ϕ is suitable as the interpretation of a complete discourse only if it is a well-formed root formula:

(9) An expression ϕ is a **well-formed root formula** iff:

a. ϕ is a formula;

b. If ${}^X\psi$ occurs anywhere in ϕ, then $\mathcal{A}(X) = \psi$;

c. If X is retrieved anywhere in ϕ, then X is stored somewhere in ϕ;

d. There are no circular dependencies among formula labels.

Label X *immediately depends on* label Y if Y is retrieved in the formula $\mathcal{A}(X)$, and *depends on* is the transitive closure. The semantic value of expression ϕ is defined in Figure 2.

(10) Term denotations are pluralities (sets of atoms):

a.	$[\![x]\!]^g = [\![[x]]\!]^g$	$= g(x)$	[simple terms]
b.	$[\![\Sigma_x\phi]\!]^g$	$= \bigcup\{h(x) : C_{L_\phi}\phi(g,h)\}$	[summation terms]

(11) Formula denotations are 1 or 0:

a.	$[\![\textsc{p}(\tau_1,\tau_2,\ldots)]\!]^g$	$= 1$ iff $\langle[\![\tau_1]\!]^g,[\![\tau_2]\!]^g,\ldots\rangle\in\mathcal{M}(\textsc{p})$	[predication]
b.	$[\![\tau_1 = \tau_2]\!]^g$	$= 1$ iff $[\![\tau_1]\!]^g = [\![\tau_2]\!]^g$	[equality]
c.	$[\![\neg\phi]\!]^g$	$= 1$ iff $[\![\phi]\!]^g = 0$	[negation]
d.	$[\![\phi \wedge \psi]\!]^g$	$= 1$ iff $[\![\phi]\!]^g{=}[\![\psi]\!]^g{=}1$	[conjunction]
e.	$[\![\exists\phi]\!]^g$	$= 1$ iff $\exists h\,.\,C_{L_\phi}\phi(g,h)$	[existential closure]
f.	$[\![{}^X\phi]\!]^g$	$= [\![\phi]\!]^g$	[formula storage]
g.	$[\![{}_X\phi]\!]^g$	$= [\![\mathcal{A}(X) \wedge \phi]\!]^g$	[formula retrieval]

(12) Local variables are bracketed variables not embedded in a subblock:

a.	$L_{[x]}$	$= \{x\}$
b.	L_x	$= \varnothing$
c.	$L_{\Sigma_x\phi} = L_{\exists\phi}$	$= \varnothing$
d.	$L_{\textsc{p}(\tau_1,\tau_2,\ldots)}$	$= L_{\tau_1} \cup L_{\tau_2} \cup \ldots$
e.	$L_{(\phi\wedge\psi)}$	$= L_\phi \cup L_\psi$
f.	$L_{{}^X\phi} = L_{\neg\phi}$	$= L_\phi$
g.	$L_{{}_X\phi}$	$= L_{(\mathcal{A}(X)\wedge\phi)}$

Figure 2: Formal Specification of FOL-PLUS

In brief, FOL-PLUS is FOL with the addition of summation terms (10b) and labeled subformulas (11f) & (11g). Summation terms $\Sigma_x\phi$ (*i*) perform a closure of local variables via C_{L_ϕ}, and then (*ii*) denote the union of all values for x that satisfy ϕ. Formula "storage" is implemented as an identity check in the second clause of the definition of well-formedness (9); thus $[\![{}^X\phi]\!]^g$ is simply $[\![\phi]\!]^g$. A formula retrieval expression ${}_X\phi$ denotes the conjunction of the stored formula $\mathcal{A}(X)$ with the current formula ϕ. Note that ${}^X_Y\phi$ is an abbreviation for ${}^X({}_Y\phi)$, inasmuch as ${}_Y({}^X\phi)$ is syntactically ill-formed.

Otherwise, the definitions for FOL-PLUS are identical to those of FOL, with the exception that existential closure (11e) is unselective in FOL-PLUS. Instead of closing a named set of variables, FOL-PLUS existentials $\ulcorner\exists\phi\urcorner$ close all of ϕ's **local variables** L_ϕ. A full recursive definition of L_ϕ is given in (12), but, in a word, the local variables of ϕ are the variables that occur bracketed in ϕ, where "in ϕ" means not embedded in a subblock introduced by $\exists$ or Σ. Free variables of ϕ that are not local variables are called **external variables.**

To reduce clutter, we will henceforth assume that the variables to be closed by C are the local variables of its complement, if not otherwise specified:

(13) $C\phi$ is shorthand for $C_{L_\phi}\phi$.

For convenience, the remaining common operators may be defined in the usual way. As defined operators, they are nonessential:

(14) a. $\phi \vee \psi$ is shorthand for $\neg(\neg\phi \wedge \neg\psi)$
b. $\phi \rightarrow \psi$ is shorthand for $\neg(\phi \wedge \neg\psi)$
c. $\forall\phi$ is shorthand for $\neg\exists\neg\phi$
d. $\sim\phi$ is shorthand for $\neg\exists\phi$

"Dynamic" negation (14d) is less familiar than the others, but is widely used in the dynamic logic literature.

1.4 Discussion

FOL-PLUS is not without antecedents. Irene Heim's (1982) dissertation proposes treating indefinites as variables, whose quantificational force comes from higher operators. The dynamic semantics she introduces (Chapter 3) is particularly well known, but she also presents a static system (Chapter 2) in which sentences, and complete discourses, essentially denote propositions, returning truth values relative to a contextual assignment of values to indices. Following Lewis (1975), Heim even describes the need to distinguish local and external variables within each operator's block (Heim calls these blocks "operator-headed molecular formulas"). In particular, Heim's operators close the local variables (those introduced by indefinites) in their scope, while allowing non-local variables to essentially scope out. FOL-PLUS is partly inspired by that system.

Further, the FOL-PLUS summation operator bears similarities to an *Abstraction* operator that Kamp and Reyle (1993) propose, which returns a set of individuals which satisfy a formula when substituted for a particular variable, i.e., $\{x' : \phi[x{\mapsto}x']\}$. And the idea of labeled subformulas is adopted outright from Keshet (2018), who proposes *update variables* that store and retrieve the meanings of subformulas in a larger formula. What is special about FOL-PLUS is the way it combines these features into a system that is radically simpler than previous systems, without any sacrifice of empirical coverage.

Let us examine each of the distinguishing features of FOL-PLUS in more detail, after which we turn to the empirical coverage of the system.

1.4.1 Unselective quantification

In FOL-PLUS, the free variables of an open formula are divided into local and external variables; the former, but not the latter, are marked for closure. For example, in (15a), both x and y are free, but x is local (that is, bracketed) whereas y is external. Closure operators like $\exists$ in FOL-PLUS are **unselective** (Lewis 1975; Heim 1982), in that they close all local variables in their scope block. Thus, (15b) has the same interpretation as the FOL formula $\ulcorner \exists x\, (\textsc{dog}(x) \wedge \textsc{saw}(x, y)) \urcorner$. And just like this FOL formula, in (15b), only y is free, and it is again external. In (15c), only y is free, and it is local. Note that the properties *free, local,* and *external* apply to variables with respect to a containing expression, not to individual variable occurrences.

(15) a. $\textsc{dog}([x]) \wedge \textsc{saw}(x, y)$
b. $\exists\, (\textsc{dog}([x]) \wedge \textsc{saw}(x, y))$
c. $\exists\, (\textsc{dog}([x]) \wedge \textsc{saw}(x, y)) \wedge \textsc{cat}([y])$

Incidentally, when ϕ contains no local variables, $\exists\phi$ is equivalent to ϕ: for example, $\exists\, \textsc{dog}(x)$ is equivalent to $\textsc{dog}(x)$.

FOL-PLUS does not forbid cataphora: the expression $\ulcorner \textsc{p}([x]) \wedge \textsc{q}(x)\urcorner$ has exactly the same interpretation as $\ulcorner \textsc{q}(x) \wedge \textsc{p}([x])\urcorner$. Bracketing a variable a second time in the same block is also not forbidden, though it has no additional effect. Let us say that an expression is in **standard form** if it satisfies the following syntactic conditions:[2]

(16) a. In a given block, if a variable occurs in bracketed form, then every unbracketed occurrence of the same variable follows the bracketed occurrence.
b. No variable is bracketed multiple times in the same block.

Any FOL-PLUS expression can be converted to standard form without changing its meaning, by moving brackets to the front of the block. To convert an expression ϕ to standard form, for each block in ϕ, let L be the local variables of the block. Delete brackets from all bracketed variable occurrences in the block, then, where $L = \{x, y, \ldots\}$, replace the root expression ψ of the block with:

(17) $\mathbb{T}([x]) \wedge \mathbb{T}([y]) \wedge \ldots \wedge \psi$

where $\mathbb{T}$ is a predicate that returns constant true (concretely, we may define $\mathbb{T}(\tau) \triangleq \tau{=}\tau$). For example, (18a) becomes (18b):

(18) a. $\textsc{p}(x) \wedge \textsc{q}([x], \sum_y (\textsc{r}(y, [y]) \wedge \textsc{s}(x, [y])))$
b. $\mathbb{T}([x]) \wedge \textsc{p}(x) \wedge \textsc{q}(x, \sum_y (\mathbb{T}([y]) \wedge \textsc{r}(y, y) \wedge \textsc{s}(x, y)))$

Conversion may also be done recursively, in which case an expression $\mathbb{T}(x)$ may arise with an unbracketed variable x; any such expression may be deleted.

1.4.2 Summation terms

Generalized quantifiers are captured in FOL-PLUS via summation terms:

(19) $\textsc{every}\,(\Sigma_x \textsc{person}([x]), \Sigma_x(\textsc{person}([x]) \wedge \textsc{person}([y]) \wedge \textsc{needs}(x, y)))$
"Everybody needs somebody"

In (19), $\ulcorner \Sigma_x \textsc{person}([x])\urcorner$ denotes the union of every value for x that satisfies the predicate $\textsc{person}$, i.e., everybody. Likewise, $\ulcorner \Sigma_x(\textsc{person}([x]) \wedge \textsc{person}([y]) \wedge \textsc{needs}(x, y))\urcorner$ denotes $\bigcup\{x : \exists y(\textsc{person}(x) \wedge \textsc{person}(y) \wedge \textsc{needs}(x, y))\}$, i.e.,

[2]Where C is a closure operator, note that a block $C_X\phi$ that incorporates the formula labeled X by reference is considered to contain any bracketed variable that occurs free in $\mathcal{A}(X)$.

everybody who needs somebody.[3]

1.4.3 Labeled subformulas

Following the DUAL system of Keshet (2018), FOL-PLUS allows labeling of subformulas for later reference. In fact, this feature can simplify the analysis of generalized quantifiers, as shown in (20), which is equivalent to (19):

(20) $\textsc{every}\left(\Sigma_x {}^{X}(\textsc{person}([x]))\,, \Sigma_x {}^{Y}_{X}(\textsc{person}([y]) \wedge \textsc{needs}(x,y))\right)$
"Everybody needs somebody"

Here, the restrictor formula $\ulcorner {}^{X}(\textsc{person}([x]))\urcorner$ assigns the label X to the subformula $\ulcorner\textsc{person}([x])\urcorner$. The subscript X on the nuclear scope formula then retrieves and incorporates the formula with label X.[4]

The pattern ${}^{X_1}\phi_1 \ldots {}^{X_2}_{X_1}\phi_2 \ldots {}^{X_3}_{X_2}\phi_3 \ldots$ is common. One may think of each segment ${}^{X_i}_{X_{i-1}}\phi_i$ as a continuation that "resumes the thread" of the one before, and for that reason we call it **block threading**. We observe that, in the case of generalized quantifiers, conservativity is an immediate consequence of block threading. This hardly constitutes an explanation for conservativity, though it

[3]To more rigorously derive the paraphrases just given, consider the following, in which "$\ldots[x]\ldots[y]\ldots z\ldots$" stands for an arbitrary formula in which x and y represent local variables and z is a representative external variable:

(i) $[\![\Sigma_x(\ldots[x]\ldots[y]\ldots z\ldots)]\!]^g$

a. $= \bigcup\{h(x) : C(\ldots[x]\ldots[y]\ldots z\ldots)(g,h)\}$

b. $= \bigcup\{h(x) : g[x,y]h \wedge [\![\ldots[x]\ldots[y]\ldots z\ldots]\!]^h = 1\}$

c. $= \bigcup\{u : \exists h(h(x) = u \wedge g[x,y]h \wedge (\ldots h(x)\ldots h(y)\ldots h(z)\ldots))\}$

d. $= \bigcup\{u : \exists h(g[y]h \wedge (\ldots u\ldots h(y)\ldots g(z)\ldots))\}$

e. $= \bigcup\{u : \exists v(\ldots u\ldots v\ldots g(z)\ldots)\}$

f. $= \bigcup\{x : \exists y(\ldots x\ldots y\ldots g(z)\ldots)\}$

g. $= [\![\bigcup\{x : \exists y(\ldots x\ldots y\ldots z\ldots)\}]\!]^g$

In short, we can paraphrase a Σ_x expression with the union of a standard set abstraction, provided that (1) we existentially close all local variables other than the variable of summation, and (2) we leave external variables free, to pick up their values from the context of evaluation. In particular:

(ii) a. $\Sigma_x\,\textsc{person}([x]) = \bigcup\{x : \textsc{person}(x)\}$

b. $\Sigma_x\,(\textsc{person}([x]) \wedge \textsc{person}([y]) \wedge \textsc{needs}(x,y))$
$= \bigcup\{x : \exists y\,(\textsc{person}(x) \wedge \textsc{person}(y) \wedge \textsc{needs}(x,y))\}$

[4]For simplicity, we use expressions of FOL-PLUS as values for formula labels, but if one prefers values that are not bound to the particular syntax of FOL-PLUS, it is possible to do so. Let us define the **import** of an expression ϕ to include not only its semantic-value function $[\![\phi]\!] \triangleq \lambda g\,.\,[\![\phi]\!]^g$, but also its local variables:

(i) $\text{import}(\phi) \triangleq ([\![\phi]\!]\,, L_\phi)$

As an alternative to using ϕ itself as $\mathcal{A}(X)$, one may use the import of ϕ. Two expressions are **fully equivalent** if and only if they have the same import. Having the same truth function is necessary but not sufficient. Among other things, two expressions are interchangeable as the argument of C if and only if they are fully equivalent.

does make it a special case of a more general pattern.

2 Empirical Results

In this section, we present our main empirical results. In keeping with practice in the dynamic semantics literature, we often give the FOL-PLUS formula for natural-language sentences, but to facilitate comparison to the E-type literature, we also provide a formal translation function from natural language parse trees to FOL-PLUS, in the style of Heim and Kratzer (1998). We will show that, despite its simplicity, FOL-PLUS covers all major improper scope phenomena at least as well as alternative approaches, whether E-type or dynamic.

2.1 A Compositional Translation

For familiarity, and ease of comparison to E-type proposals, we will interpret natural language expressions in a system as close as possible to the standard Heim and Kratzer (1998) system, with the main substantive difference being the use of FOL-PLUS, instead of English, as a meta-language. This is quite different from the usual approach in the plural dynamic semantics literature (e.g. Brasoveanu 2007), which builds on the **Compositional DRT** system due to Muskens (1996). Our hope is that combining a familiar logic (FOL) with a familiar interpretation procedure (Heim and Kratzer 1998) will itself contribute to our comparison of various approaches to improper scope phenomena.

To aid in this project, we will first need to introduce lambda expressions into FOL-PLUS. We do so only as a temporary expedient, though—we will design the translation so that all lambda expressions are eliminated when the output expression is simplified. Although the lambda expressions themselves are rather intuitive, the technical details are somewhat complex. Please see Appendix A for details, but the two major items that allow lambda functions in FOL-PLUS are (i) the conversion of every expression to standard form before β-reduction, and (ii) the liberal use of fresh variables, which we will conventionally notate using (decorated versions of) the variable z.

Next, we will use a translation function $'$ that takes a tree α as input and returns an expression α' of FOL-PLUS as output. The translation from α to α' defines the (Heim & Kratzer-style) semantic value of α indirectly via the definition in (21). This allows us to define semantic types for natural-language and FOL-PLUS expressions as usual: e for individuals, t for truth values, and $\langle\sigma,\tau\rangle$ for functions from σ to τ. Finally, we will often use the "squiggly arrow" notation shown in (22) to improve readability, especially when translating a larger phrase or sentence.

(21) $[\![\alpha]\!]^g \quad \triangleq \quad [\![\alpha']\!]^g$

(22) $\alpha \rightsquigarrow \beta \quad \triangleq \quad \alpha' = \beta$

2.1.1 Rules of Interpretation

With (21) in place, all of the standard Heim & Kratzer rules can be restated as rules translating from natural language into FOL-PLUS:

(23) **Traces and Pronouns (TP)**: $\alpha_x \rightsquigarrow x$ and $\alpha^x \rightsquigarrow [x]$
for any trace or pronoun α and lowercase index x.
Terminal Nodes (TN): α' is listed in the lexicon
for other lexical terminal nodes α.
Non-branching Nodes (NN): $[_\alpha\, \beta\,] \rightsquigarrow \beta'$
for any non-branching node α whose child is β.
Functional Application (FA): $[_\alpha\, \beta\, \gamma\,] \rightsquigarrow \beta'(\gamma')$ and $[_\alpha\, \gamma\, \beta\,] \rightsquigarrow \beta'(\gamma')$
for any branching node whose children are β of type $\langle\sigma, \tau\rangle$ and γ of type σ, where β is not a closure operator.
Predicate Modification: $[_\alpha\, \beta\, \gamma\,] \rightsquigarrow \lambda z\ .\ \beta'(z) \wedge \gamma'(z)$
for any branching node α with children β and γ both of type $\langle e, t\rangle$, where z is a fresh variable.
Predicate Abstraction (PA): $[_\alpha\, \lambda_z\, \beta\,] \rightsquigarrow \lambda z\ .\ \beta'$
for any branching node α whose children are a relative operator λ indexed z and β of type t, where z is a fresh variable.

Notice that the rules above assume lowercase syntactic indices which correspond to variables in FOL-PLUS. Superscript variables on pronouns and traces translate to bracketed variables and subscripts to unbracketed variables.[5]

Next, we add two new rules, the second of which assumes uppercase, prefixed syntactic indices corresponding to formula labels in FOL-PLUS:

(24) **Discourse Conjunction (DC)**: $[_\alpha\, \beta\ \gamma\,] \rightsquigarrow \beta' \wedge \gamma'$
for any discourse α with children β and γ, both of type t.
Discourse Functional Application (DFA): $[_\alpha\, \beta_x\, {}^X_Y\gamma\,] \rightsquigarrow \beta'_x\left({}^X_Y\gamma'\right)$
for any branching node α whose children are a closure operator β with optional lowercase index x and γ of type t with optional uppercase indices X and Y.

Discourse Conjunction merely codifies assumptions already present in prior (static semantics) literature, extending the Heim & Kratzer system to interpret multi-sentence discourses. It can be conceived of as a generalization of Predicate Modification to the zero-place predicate (type t) denotations of sentences and discourses, which are interpreted via simple conjunction (cf. Keenan and Faltz 1978; Gazdar 1980; Rooth and Partee 1983). Discourse Functional Application is essentially a special case of functional application (FA). We list it separately as it deals with the subscripts and superscripts on the closure operator and its argument. DFA is the only point at which formula-label storage and retrieval is introduced. Formula-label storage and retrieval arguably formalizes a type of anaphora that has been posited, for instance, to connect phrases to their

[5]This, and the superscript formula labels, follow a convention due to Barwise (1987) whereby superscript indices mark antecedents.

focus antecedents (Rooth 1992; Schwarzschild 1999). It can also be viewed as a variety of ellipsis, another process that has been proposed for E-type anaphora (Elbourne 2005).

2.1.2 Lexical Entries

Some illustrative lexical entries are given below, where z, z_1, z_2, etc. are always fresh variables:

(25) a. **Closure operators**: $\mathbf{S}_x \rightsquigarrow \Sigma_x$, $\mathbf{E} \rightsquigarrow \exists$, $\mathbf{N} \rightsquigarrow \sim$
b. **Indefinite determiners**: $\text{a}^x \rightsquigarrow \lambda z_1 \lambda z_2 \,.\, z_1([x]) \wedge z_2(x)$
Same for *any*^{x}, *some*^{x}, etc.
c. **Common Ns, As, Vs**: $\text{dog} \rightsquigarrow \lambda z \,.\, \textsc{dog}(z)$
Same for *brown*, *bark*, etc.
d. **Transitive items**: $\text{saw} \rightsquigarrow \lambda z_1 \lambda z_2 \,.\, \textsc{saw}(z_2, z_1)$
Same for *ate*, *fond*, *in*, etc.
e. **Quantificational Ds**: $\text{most} \rightsquigarrow \lambda z_1 \lambda z_2 \,.\, \textsc{most}(z_1, z_2)$
Same for *every*, *few*, *exactly nine*, etc.
f. **Empty NPs**: $\langle\rangle \rightsquigarrow \mathbb{T}$ (constant true)

Finally:

(26) Unless otherwise specified, both uppercase and lowercase indices may be added to any node without changing its interpretation.

2.2 Some basic examples

Simple pronouns and traces translate as variables:

(27) a. $\text{it}_d \text{ barked} \rightsquigarrow \text{barked}'(\text{it}_d{}') = \textsc{barked}(\text{d})$ [FA]
b. $t_d \text{ barked} \rightsquigarrow \textsc{barked}(d)$ [FA]

The treatment of noun modification and relative clauses is adopted unchanged from Heim and Kratzer:

(28) a. $\text{that}_z \text{ barked} \rightsquigarrow \lambda z(\textsc{barked}(z))$ [PA]
b. dog [that$_{z_1}$ barked]
$\rightsquigarrow \lambda z_2(\textsc{dog}(z_2) \wedge \lambda z_1(\textsc{barked}(z_1))(z_2))$ [PM]
$\overset{\beta}{\Rightarrow} \lambda z_2(\textsc{dog}(z_2) \wedge \textsc{barked}(z_2))$

The syntactic λ operator that is introduced by quantifier raising (QR) behaves exactly like a relative pronoun:

(29) $\lambda_z \; [t_z \text{ barked}] \rightsquigarrow \lambda z(\textsc{barked}(z))$ [PA]

Indefinites are treated syntactically as generalized quantifiers, albeit rather simple ones; see (25b). Following standard assumptions, they raise via QR to take scope over the clause core, and this movement introduces a syntactic λ

operator. As mentioned above, we assume a fresh variable for this λ and convert to standard form before β-reduction.

(30) [a^d dog] barked
$\overset{\text{QR}}{\Rightarrow}$ [a^d dog] [λ_{z_1} [t_{z_1} barked]]
$\rightsquigarrow (\lambda z_2 \lambda z_3 \,.\, z_2([d]) \wedge z_3(d))(\textsc{dog})(\lambda z_1\, \textsc{barked}(z_1))$
$\overset{\text{std}}{\Rightarrow} \mathbb{T}([d]) \wedge (\lambda z_2 \lambda z_3 \,.\, z_2(d) \wedge z_3(d))(\textsc{dog})(\lambda z_1\, \textsc{barked}(z_1))$
$\overset{\beta}{\Rightarrow} \mathbb{T}([d]) \wedge \textsc{dog}(d) \wedge \textsc{barked}(d)$

An example with two indefinites:

(31) [a^f farmer] owns [a^d donkey]
$\overset{\text{QR}}{\Rightarrow}$ [a^f farmer] λ_{z_1} [[a^d donkey] λ_{z_2} [t_{z_1} owns t_{z_2}]]

The portion beginning with [a^d donkey] translates just as (30), yielding (32a). (The transitive verb introduces additional lambdas, but they are eliminated by β-reduction.) The pattern is then repeated to combine [a^f farmer] with its arguments, yielding (32b).

(32) a. [[a^d donkey] λ_{z_2} [t_{z_1} owns t_{z_2}]]
$\Rightarrow \mathbb{T}([d]) \wedge \textsc{donkey}(d) \wedge \textsc{owns}(z_1, d)$
b. [[a^f farmer] λ_{z_1} [[a^d donkey] λ_{z_2} [t_{z_1} owns t_{z_2}]]]
$\rightsquigarrow (\lambda z_2 \lambda z_3 \,.\, z_2([f]) \wedge z_3(f))(\textsc{farmer})$
$(\lambda z_1 \,.\, \mathbb{T}([d]) \wedge \textsc{donkey}(d) \wedge \textsc{owns}(z_1, d))$
$\overset{\text{std}}{\Rightarrow} \mathbb{T}([f]) \wedge \mathbb{T}([d]) \wedge (\lambda z_2 \lambda z_3 \,.\, z_2(f) \wedge z_3(f))(\textsc{farmer})$
$(\lambda z_1 \,.\, \textsc{donkey}(d) \wedge \textsc{owns}(z_1, d))$
$\overset{\beta}{\Rightarrow} \mathbb{T}([f]) \wedge \mathbb{T}([d]) \wedge \textsc{farmer}(f) \wedge \textsc{donkey}(d) \wedge \textsc{owns}(f, d)$

As a final simplification step, expressions $\mathbb{T}([x])$ can be eliminated by moving the brackets to the next occurrence of the same variable in the same block, and we will generally do so for the sake of readability; e.g., (32) becomes $\textsc{farmer}([f]) \wedge \textsc{donkey}([d]) \wedge \textsc{owns}(f, d)$.

2.3 Discourse interpretation

We follow Heim (1982) in assuming that multiple sentences of a discourse form a single interpretable structure, with an existential closure operator (for us, **E**) taking scope over the entire discourse. A discourse may be extended without limit by adjoining sentences, but to determine a truth value for the final result, one must "cap" it with a **E** operator:

(33) $\Big[\mathbf{E}\Big[\,[\,[\mathrm{S}_0\ \mathrm{S}_1]\,\mathrm{S}_2]\ldots\mathrm{S}_n\Big]\Big]$

The sentences in (33) are combined via Discourse Conjunction, with the final **E** operator interpreted using the Discourse Functional Application rule. This is essential for distinguishing the interpretation of an indefinite from a pronoun:

(34) a. She$_x$ saw him$_y$ $\rightsquigarrow$ $\textsc{saw}(x, y)$
b. She$_x$ saw someone^y $\rightsquigarrow$ $\textsc{saw}(x, [y])$

Recall that these two formulas have identical truth conditions, since brackets do not directly affect the interpretations of variables. And yet, the sentence in (34b), where y is introduced by an indefinite, is felicitous without any previous mention of individual y; but (34a), where y is used solely as a pronoun index, is only felicitous when a referent for y has been established, whether in previous discourse, by pointing, etc. (Both sentences require previous mention of x.)

Once **E** is added, though, we actually do get different truth conditions for the two sentences (as reflected in the differing FOL paraphrases):

(35) a. **E** She$_x$ saw someone^y $\rightsquigarrow \exists(\textsc{saw}(x, [y])) \qquad \equiv \exists y\, (\textsc{saw}(x, y))$
b. **E** She$_x$ saw him$_y$ $\rightsquigarrow \exists(\textsc{saw}(x, y)) \qquad \equiv \textsc{saw}(x, y)$

The differences in meaning bear on felicity, as well. Any variables external to the discourse as a whole must be assigned felicitous values by the contextual assignment g that is used to interpret the discourse, if the discourse is to receive an interpretation. In the formalization of Figure 2, we assumed, in keeping with established practice, that assignments are total functions over the countably infinite set of variables. For the purposes of an account of felicity, one may either follow Heim and assume that assignments are partial functions, or, equivalently, designate one or more values as "non-values" such as $\star$, and deem the value of a variable x to be undefined if $g(x)$ is $\star$. We assume that, unless the value of a variable is determined by the non-linguistic context of the discourse, its value is undefined in the contextual assignment used to interpret the discourse. We can then summarize the truth and felicity conditions of a discourse as follows:

(36) A discourse [**E** α] is only felicitous relative to a contextual assignment g when $g(x)$ is not undefined, for all external variables x of α. If felicitous, the discourse is true iff $[\![\mathbf{E}\ \alpha]\!]^g$=1.

If one uses partial assignments, a discourse ϕ is felicitous relative to g only if $[\![\phi]\!]^g$ is defined.

2.4 Summation Pronouns

We turn now to summation pronouns, illustrated in (37):

(37) Everyone λ_z [$_{\text{S}}$ t^z [$_{\text{VP}}$ brought a^p present]]^B. They are on that table.

Standard E-type analyses take pronouns such as *they* in (37) to be complex: they apply a function f to one or more arguments of type e. For instance, in cases like (37), Heim (1990, p.172) suggests an analysis like (38). A salient function (f below) relates groups of individuals to the presents they brought. When this function is applied to an implicit free variable (a below) denoting a salient set of partygoers, we derive the desired denotation:

(38) $[\![\text{they}]\!]^g = [\![[f\ a]]\!]^g$, where perhaps

a. $g(a) =$ a salient group of partygoers

b. $g(f) = [\lambda x$. the presents that x brought$]$

There is a natural implementation, parallel to Heim's, in terms of summation. Consider the FOL-PLUS expression given in (39a), that translates the S node in (37). The desired interpretation for *they* is given in (39b), namely, the sum of values for p across assignments that satisfy (39a), which is paraphrased in FOL as (39c).

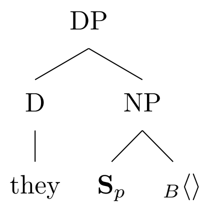
(39) a. S $\rightsquigarrow \textsc{present}([p]) \wedge \textsc{brought}([z], p)$

b. they $\rightsquigarrow \Sigma_p\,(\textsc{present}([p]) \wedge \textsc{brought}([z], p))$

c. $\equiv \bigcup\{p : \exists z\,(\textsc{present}(p) \wedge \textsc{brought}(z, p))\}$

We obtain the desired translation (39b) by assuming the syntactic structure (40) for this variety of pronoun, similar to the structure of E-type complex pronouns.

(40)



We call pronouns with this structure **summation pronouns.** In (40), $\mathbf{S}_p$ is a syntactic closure operator, interpreted via Discourse Functional Application, that translates to Σ_p (25a), and $_B\langle\rangle$ is an empty NP that translates to a tautological constant (25f). The subscript formula label B does not affect the interpretation of $_B\langle\rangle$, but is incorporated by DFA into the translation of the intermediate node:

(41) $[\mathbf{S}_p\ {}_B\langle\rangle] \rightsquigarrow \Sigma_{pB}\mathbb{T}$.

There is an anaphoric dependency on a previous node in the discourse, namely the S in (37). The formula-label index B on this S ensures that $\mathcal{A}(B) = \ulcorner\textsc{present}([p]) \wedge \textsc{brought}([z], p)\urcorner$. The index B on $\langle\rangle$ retrieves this formula:

(42) $\Sigma_{pB}\mathbb{T} = \Sigma_p(\textsc{present}([p]) \wedge \textsc{brought}([z], p))$

We assume that the overt pronoun *they* itself contributes little to the denotation: perhaps simply number and gender presuppositions, as well as an existential presupposition to be discussed below.

Recall that even inside closed blocks, the values for external variables come from the contextual assignment g. This is motivated by examples like (43):

(43) Everyone $\lambda_z\ {}^G[t^z$ gave her$_y$ a^p present$]$. [They $\mathbf{S}_p\ {}_G\langle\rangle$] are on that table.

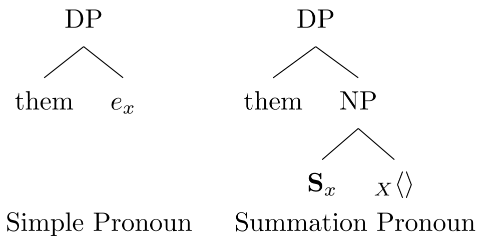
a. $\mathcal{A}(G) = \textsc{present}([p]) \wedge \textsc{gave}([z], y, p)$

b. $$\begin{aligned}[\![\mathbf{S}_{p\ G}\langle\rangle]\!]^g &= \bigcup\{h(p) : (C\mathcal{A}(G))(g,h)\} \\ &= \bigcup\{h(p) : C\,(\textsc{present}([p]) \wedge \textsc{gave}([z], y, p))\,(g,h)\} \\ &\equiv \bigcup\{p : \exists z\,(\textsc{present}(p) \wedge \textsc{gave}(x, g(y), p))\}\end{aligned}$$

Here, the pronoun *her*$_y$ represents an external variable, established either elsewhere in the discourse or by the extra-linguistic context. Rather than being existentially closed in the definition of *they*, like the local variable z, the external variable y takes its value from the context: note the $g(y)$ in the last line of (43b), which is a FOL paraphrase. The set of presents denoted by *they* varies with the individual y denoted by *her.*

Discussion. We now have two kinds of pronominal DPs: simple pronouns, marked with a lowercase subscript variable, and summation pronouns, combining with a summation operator. To give these two a parallel analysis, we could analyze simple pronouns as combining with an empty pronominal element e_x:

(44)



Under this view, it is possible to define the lexical pronoun *them* consistently across both types. The complements of this D are e_x and $[\mathbf{S}_x\ {}_X\langle\rangle]$, which translate as x and $\Sigma_x({}_X\mathbb{T})$, both of which denote pluralities. Thus, by FA, a pronoun is a function of type $\langle e, e\rangle$. To give a few examples:

(45) a. $[\![\text{her}]\!] = [\lambda z : |z|{=}1 \wedge \textsc{female}(z)\ .\ z]$
b. $[\![\text{it}]\!] = [\lambda z : |z|{=}1 \wedge \textsc{inanimate}(z)\ .\ z]$
c. $[\![\text{them}]\!] = [\lambda z : |z|{>}1\ .\ z]$

The major difference between the two pronoun types is that summation pronouns are exhaustive, referring to a certain set of individuals defined earlier, while simple pronouns vary in value according to the assignment. This difference is illustrated in (46), after an example in Evans (1977), where two different second sentences bring out the two readings of the same pronoun. As Evans notes, *them* in (46a) is most easily heard as an exhaustive pronoun, comprising all of John's sheep. This, then, is a summation pronoun, deriving its value from a formula label stored by the previous sentence. But *they* in (46b) clearly denotes a subset of John's sheep; in this case, we have a simple pronoun.

(46) John owns some sheep down on his farm.
a. His neighbor Harry vaccinates them in the spring.
b. They got out last night while the rest of his sheep were sleeping.

Another important difference between the types is that antecedents to simple pronouns are strictly constrained to their containing block, while summation pronouns may make use of formula labels defined in a deeply embedded position. For instance, when the variable in question is local to an embedded block, only the exhaustive, summation reading (47a) is available, as predicted:

(47) Every farmer around here owns some sheep.
  a. Harry vaccinates them in the spring.
  b. #They got out while the other sheep in their flock were sleeping.

In fact, formula-label anaphora is so free that it allows reference out of typically inaccessible contexts, such as negation. This freedom helps explain cases that traditional dynamic systems struggle with, such as double negation:

(48) a. It's not **N** like he$_x$ doesn't **N** $^O[t_x$ own a^c car].
  b. [It $\mathbf{S}_{c\ O}\langle\rangle$]'s just in the shop.

The pronoun *it* in (48b) is perfectly acceptable, even though its antecedent is embedded (twice) under negation. It denotes the sum of all cars that x owns. The use of the singular pronoun presupposes that this sum is a singleton—i.e., x owns one car—which is easily accommodated. Despite this flexibility, the system also restricts anaphora out of negation where it should be restricted:

(49) a. He$_x$ doesn't **N** $^O[t_x$ own a^c car].
  b. #[It $\mathbf{S}_{c\ O}\langle\rangle$]'s in his garage.

Here, (49a) asserts that x doesn't own a car, and therefore the presupposition of the pronoun cannot be met: if x owns no cars, there is no way for the sum of all his cars to be a singleton; it will instead be empty.[6]

Finally, formula-label anaphora can explain so-called wide-scope, or referential, indefinites (Fodor and Sag 1982), which seem to scope out of an apparent scope island. One class of explanations for wide-scope indefinites takes them to simply denote a value that does not vary across the values of a higher quantifier (Reinhart 1997; Winter 1997; Kratzer and Shimoyama 2002; Schwarzschild 2002; Brasoveanu and Farkas 2011). For instance, *she* in (50) seems to refer to a single teacher; and yet, the apparent antecedent is within a scope island:

(50) Everyone $^B[t^z$ believes that a^t teacher from my school is better than theirs]. [She $\mathbf{S}_{t\ B}\langle\rangle$] is rather talented.

(51) $\mathbf{S}_{tB}\langle\rangle \rightsquigarrow \Sigma_t({}_B\mathbb{T}) = \Sigma_t\left(\textsc{teacher}([t]) \wedge \textsc{believes-better}([z], t)\right)$

[6]Similar effects can be seen in downward entailing environments, as shown in (i). The first clause of (i) is compatible with there being no congressmen who admire Kennedy, although its most salient reading is strengthened by an implicature excluding this case. Notice that the existence presupposition of the pronoun *they* in the second clause also requires there to be (multiple, in fact) congressmen who admire Kennedy. Either the strengthened meaning of (i) is assumed, or else this presupposition is accommodated to allow the use of the pronoun.

(i) Few congressmen admire Kennedy, and they are very junior. (Evans 1980)

The value in (51) is the set of teachers that someone believes to be better than their own. If it just so happens that everyone thinks the same teacher is better than their own, then that set will indeed be a singleton. Under such an explanation, the singular marking on *she* arises because it is a singleton, and yet everything else is the same as in the plural cases.

2.5 Paycheck Pronouns

Summation pronouns can also handle **paycheck pronouns** (Karttunen 1969), so-called because of examples like (52).

(52) The woman who saved her paycheck was wiser than the woman who spent it. (cf. Jacobson 2000)

Although paycheck pronouns are easily accommodated within E-type approaches, they are notoriously difficult for plural dynamic logics to capture (Nouwen 2020). We handle them by using formula anaphora (again) as a replacement for E-type "salient functions." Take the case in (53), for instance. Standard plural logics only store individuals already mentioned or quantified over. So, after (53a), they would only store the dioramas that girls made and brought to class, not any other dioramas. And yet the pronoun *it* in (53b) seems to refer to dioramas left at home, new individuals not mentioned before.

(53) a. Almost every girl brought the diorama she made to class.
b. A few left it at home, though.

In our system, a summation pronoun may reference a discourse node, but evaluate it in a new context, i.e., under a new assignment (cf. Keshet 2018). Take the following structures for (53):

(54) a. Almost every girl
$\mathbf{S}_z$ t^z brought [$_{\text{DP}}$ the $\mathbf{S}_d$ ^{M}[$_{\text{NP}}$ t^d diorama she$_z$ made]]
b. A few $\mathbf{S}_z$ t^z left [it $\mathbf{S}_d$ $_M\langle\rangle$] at home.

The structure assigned to the definite DP in (54a) parallels the structure (40) assigned to summation pronouns. Its NP, with superscript M, provides the antecedent to the empty NP $_M\langle\rangle$ in the summation pronoun in (54b). But the antecedent NP contains an external variable, namely z, the quantified variable of the sentence in (54a). The meaning of $_M\langle\rangle$ in (54b) therefore also varies with the external variable z, as shown in the following denotation (shown in FOL).

(55) $[\![\text{it } \mathbf{S}_d\ _M\langle\rangle]\!]^g = \bigcup\{d : \textsc{diorama}(d) \wedge \textsc{made}(g(z), d)\}$

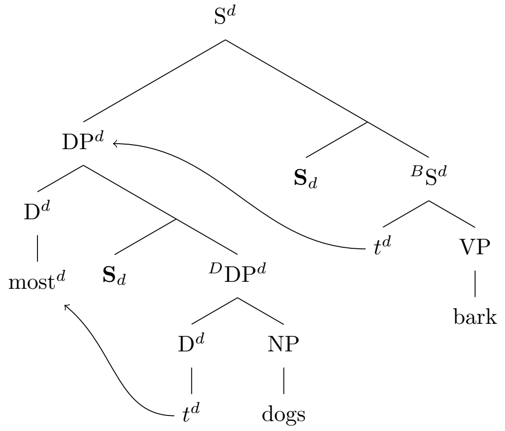
In the new context, z is bound by the quantifer of (54b), which quantifies over the few girls who forgot their dioramas. Thus, the pronoun overall refers to these forgotten dioramas rather than the ones brought to class. This is the hallmark of a paycheck pronoun.[7]

[7] Jacobson (2000) points out scenarios like the following, where a paycheck pronoun seems

2.6 Quantification

We now turn to the analysis of quantifiers, which, like summation pronouns, also employ the summation operator. In fact, in our system, quantificational determiners are simple two-place predicates of type $\langle e, \langle e, t\rangle\rangle$, both of whose arguments are created via **S** (cf. Barwise and Cooper 1981). A bit of syntactic housekeeping is required, though, to shape the arguments of a standard generalized quantifier into a form suitable for the summation operator. Consider first the following preliminary analysis, along with a meaning for *most*:

(56)



(57) most $\rightsquigarrow [\lambda r_e.\lambda s_e.\text{MOST}(r, s)] \equiv \left[\lambda r_e.\lambda s_e.|r \cap s| > \frac{1}{2}|r|\right]$

to get its antecedent from the extralinguistic context:

(i) *A new faculty member picks up her first paycheck from her mailbox. Waving it in the air, she says to a colleague:* Do most faculty members deposit it in the Credit Union? [Meaning: *Do most faculty members deposit their paycheck in the Credit Union?*]

Elbourne (2005) claims that this is easily understood in E-Type analyses, which already make frequent use of contextually supplied relations. Here, the "salient" relation must be something like "her paycheck" (or for Elbourne: "the paycheck in situation s"). But one has no way of distinguishing this from countless other relations that Elbourne would not wish to consider "salient"; see the discussion in 4.1 below. By contrast, dynamic theories (and FOL-PLUS) provide highly constrained meanings for all pronouns whose antecedents are provided by the linguistic context. We should not throw out this successful work simply because pronouns may also occasionally take antecedents from extralinguistic sources.

Instead, we propose that a gesture such as the waving of one's paycheck in (i) is incorporated into the discourse as a communicative act, contributing a meaning along the lines of (ii). That is, "look at my paycheck," just as pointing to an object might contribute the meaning *Look at that.* After all, the gesture is not successful unless the addressee (viewer) recognizes what the speaker is waving. The formula label P in (ii) is then available for subsequent anaphoric reference. We note in passing that this analysis is more straightforward in a translational semantics, versus a direct semantics like Heim & Kratzer. A logical language such as FOL-PLUS provides a common representation for meanings that spans both spoken utterances and gestures in a natural way.

(ii) $\text{ME}([x]) \wedge \text{BEHOLD}(\Sigma_p{}^P(\text{PAYCHECK-OF}([p], x)))$

We make a couple of assumptions regarding inheritance of indices:

(58) a. A trace inherits all the indices of its antecedent.
b. Indices are inherited from heads and subjects.

(58b) entails the d index on both DPs and both Ss. (The d on the lower DP and the one on the lower S are only there for consistency, but the one on the upper DP is essential, and the one on the upper S will play a role in the discussion of example (88) below.) We assume that both the D and DP undergo movement, leaving a trace and introducing a summation operator, both coindexed with the moved item. By TP, the traces are interpreted as bracketed variables, the superscript providing the variable. When the traces combine with the original NP and VP, they form nodes of type t. As complements of **S**, they constitute blocks, and we have labeled them with superscript formula labels. Finally, the **S** operators introduced by movement take the type t nodes and yield pluralities:

(59) a. $t^d \rightsquigarrow [d]$ [TP]
b. $[t^d \text{ dogs}] \rightsquigarrow \textsc{dog}([d])$ [FA]
c. $[\mathbf{S}_d\ {}^D[t^d \text{ dogs}]] \rightsquigarrow \Sigma_d{}^D\textsc{dog}([d])$ [DFA]

(60) $[\mathbf{S}_d\ {}^B[t^d \text{ bark}]] \rightsquigarrow \Sigma_d{}^B\textsc{bark}([d])$

The set (59c) is the set of dogs and (60) is the set of things that bark. The former is the *restrictor set* and the latter is the *simple scope set.* The intersection of restrictor set and simple scope set is called the *reference set* (Nouwen 2003a): here, the set of dogs that bark. A suitable predicate denotation for *most* is given, MOST, which compares the reference set to the restrictor set, deriving the correct truth conditions.

Empirical considerations lead us to refine this preliminary analysis in one respect. We have hypothesized that all and only blocks are possible antecedents for formula anaphora. And yet, it seems that the simple scope set (things that bark) is not a possible antecedent, while the reference set (dogs that bark) is:

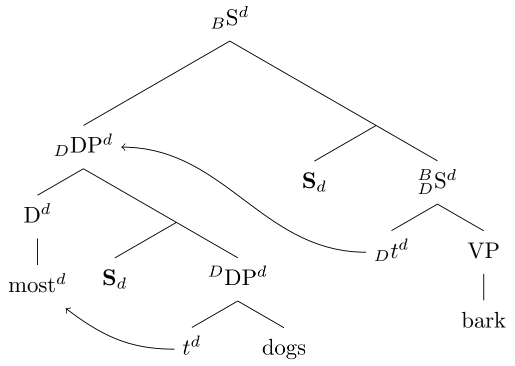
(61) Most dogs bark.
a. And most of **them** are loud, except pugs.
(=*Most dogs that bark are loud, except pugs.*)
b. #And most of **them** are loud, except seals.
(Intended: *Most things that bark are loud, except seals.*)

This pattern of facts suggests that we revise the analysis in (56) to replace the node ^{B}S, whose summation denotes the unneeded simple scope set, with a node whose summation denotes the actually available reference set. Doing so is also concordant with the postulate by Barwise and Cooper (1981) that determiners are **conservative**, in the sense that they always denote a relation between the restrictor set and the reference set, not the simple scope set.

A minimal change to structure (56), shown in (63), suffices both to capture the anaphoric facts and to assure suitable arguments for determiner relations: we add subscript D to the nuclear scope (now labeled ${}^B_D$S) in order to incorporate the meaning of the restriction into the nuclear scope. The only difference between (63) and (56) is the addition of the subscript formula labels D and B to a few nodes, triggered by the following new assumption:

(62) In the structure $[_{\text{XP}}$ X $[\mathbf{S}_x$ YP$]]$, any superscript formula label on YP is inherited as a subscript formula label on XP (overriding label inheritance from X).

(63)



Thus the upper DP obtains the subscript formula label D, which is then inherited by its trace and thence by the nuclear-scope S. (Likewise, the upper S inherits the B of the lower S as a subscript formula label. That plays no role currently, but will in the discussion of example (88).) The label D on the trace of the DP, $_Dt^d$, has no effect on its interpretation, by (26). But the labels on the nuclear-scope $^B_D\text{S}^d$ show up as labels in the FOL-PLUS translation; the result is an example of block threading (see section 1.4.3).

The denotation of MOST is simpler now. Its second argument, s, is the full reference set rather than the simple scope set:

(64) most $\rightsquigarrow [\lambda r_e.\lambda s_e.\textsc{most}(r, s)] \approx \left[\lambda r_e.\lambda s_e.|s| > \frac{1}{2}|r|\right]$

In our example, the translation of the first argument of *most* (59c) is unchanged, but a subscript D is added to the translation of the second argument (60), by DFA. The resulting translation for (63) is:

(65) $\textsc{most}(\Sigma_d{}^D\textsc{dog}([d]), \Sigma_d{}^B_D\textsc{bark}([d]))$

The first summation denotes all dogs, and the second all dogs that bark.

Again, parallel to (43) above, closure does not capture external variables. For instance, *hers*$_x$ in (66) will get its value from the external context, rather than being closed within the quantification.

(66) a. Every^f friend of hers$_x$ waved.
b. $\textsc{every}(\Sigma_f{}^F\textsc{friend-of}([f], x), \Sigma_f{}^W_F\textsc{waved}([f]))$
$\equiv \textsc{every}(\Sigma_f\textsc{fr'-of}([f], x), \Sigma_f(\textsc{fr'-of}([f], x) \wedge \textsc{waved}([f])))$

The denotations of the summations vary with the choice of value for x.

2.7 Donkey Pronouns

The structure for quantifiers presented thus far embeds the meaning of the restrictor into the nuclear scope, via an uppercase subscript. A benefit of this analysis is that it readily explains donkey pronouns (Geach 1962):

(67) Everyone who owns an umbrella brought it today.

The pronoun *it* in (67) operates differently from the summation pronouns thus far analyzed: it does not seem to denote the result of a summation, which in this case might be the set of all owned umbrellas. Instead, for each umbrella-owner x, *it* denotes whichever umbrella x brought today. This is explained, though, by the inclusion of the restrictor inside the nuclear scope. Consider the following:

(68) Every $\mathbf{S}_x\ {}^{O}[_{\text{DP}}\ t^x$ one who owns an^u umbrella]
$\quad\mathbf{S}_x\ {}^{B}_{O}[_{\text{S}}\ {}_{O}t^x$ brought it$_u$ today.]

(69) a. ${}^{O}\text{DP} \quad \rightsquigarrow \textsc{umbrella}([u]) \wedge \textsc{owns}([x], u)$
b. ${}^{B}_{O}\text{S} \quad \rightsquigarrow \textsc{brought}(x, u)$
c. $\mathbf{S}_x\, {}^{B}_{O}\text{S} \quad \rightsquigarrow \Sigma_x{}^{B}_{O}\textsc{brought}(x, u)$
$\qquad \equiv \Sigma_x{}^{B}(\textsc{umb'lla}([u]) \wedge \textsc{owns}([x], u) \wedge \textsc{bro't}(x, u))$
d. $[\![\mathbf{S}_x\, {}^{B}_{O}\text{S}]\!]^g \quad = \bigcup\{x : \exists u(\textsc{umb'lla}(u) \wedge \textsc{owns}(x, u) \wedge \textsc{bro't}(x, u))\}$

The nuclear scope ${}^{B}_{O}S$ effectively carries the denotation of its antecedent ^{O}DP, and therefore requires that u be an umbrella that x owns. ${}^{B}_{O}$S will thus be true under all assignments where u is an umbrella that x owns and brought today.

This case is most easily understood as a so-called weak donkey pronoun (Schubert and Pelletier 1989): the umbrella owners in question need not have brought all of their umbrellas for the sentence to be true. The most likely scenario (since people usually only carry one umbrella) is that even those who own more than one umbrella only brought one, and the sentence is perfectly acceptable in this scenario, as predicted. The weak donkey reading is also compatible with the less likely (but not impossible) scenario that one of the multiple-umbrella owners brought more than one umbrella, perhaps as a backup or for a friend. The weak analysis allows but does not require such a scenario.

Interestingly, summation pronouns of the form "$\mathbf{S}_u\ {}_B\langle\rangle$", as in (70), will denote all the umbrellas that were actually brought. In the most likely one-umbrella-per-person scenario, the number of these umbrellas would be equal to the number of umbrellas owners. However, in the scenario where at least one multiple-umbrella owner brought more than one umbrella, (70) asserts that all of the umbrellas are in the rack, not just one per owner. So, although only one per owner is required for the truth of (67), all brought umbrellas are included in later anaphora like (70). And this is exactly as predicted.

(70) [They $\mathbf{S}_u\ {}_B\langle\rangle$] are in that rack.

Plural Donkey Pronouns. Plural indefinites can give rise to donkey anaphora as well, as illustrated by the structure in (72) for (71):

(71) Everyone who owns any sneakers wore them today.

(72) Every $\mathbf{S}_z\ {}^{O}[_{\text{DP}}\ t^z$ one who owns any^y sneakers]
$\mathbf{S}_z\ {}^{W}_{O}[_{\text{S}}\ {}_{O}t^z$ wore them$_y$ today]

The most salient reading for (71) is one requiring sneaker-owners to wear only some of their sneakers (typically one per foot), rather than all of them. This is captured nicely by the analysis described above, with values like the following for the restrictor and nuclear scope:

(73) a. $\mathbf{S}_z\ {}^{O}\text{DP} \quad \rightsquigarrow \Sigma_z(\textsc{sneakers}([y]) \wedge \textsc{owns}([z], y))$
b. $\mathbf{S}_z\ {}^{W}_{O}\text{S} \quad \rightsquigarrow \Sigma_z(\textsc{sneakers}([y]) \wedge \textsc{owns}([z], y) \wedge \textsc{wore}(z, y))$

(73a) is the set sneaker-owners, and (73b) is the set of individuals that own and wore at least one sneaker.

And yet other plural cases have different readings. For instance, the most salient reading for (74) is that each person locked up all their (own) valuables. And (75) most plainly means that each student commented on all the response papers, their own and others':

(74) Everyone who brought any valuables locked them up.

(75) Everyone who wrote any response papers commented on them all during class.

The case (75) is not captured by most dynamic plural systems (for details see Nouwen 2003b), but it is actually simpler to capture than (74) in our system. Consider the following structure:

(76) Every $\mathbf{S}_z\ {}^{W}[_{\text{DP}}\ t^z$ one who wrote any^y papers]
$\mathbf{S}_z\ {}^{C}_{W}[_{\text{S}}\ {}_{W}t^z$ commented on [them $\mathbf{S}_y\ {}_{W}\langle\rangle$] today.]

Here we have analyzed *them* as a summation pronoun, collecting all values for y in the discourse value of ${}^{W}\text{DP}$. Since z is local, this will denote the following:

(77) $\mathbf{S}_y\ {}_{W}\langle\rangle \rightsquigarrow \Sigma_y\,(\textsc{papers}([y]) \wedge \textsc{wrote}([z], y))$

In other words, it will be all papers anyone wrote.

This is not the correct analysis for (74), though, because (74) only requires each person to lock up their own valuables, rather than everyone's valuables. Consider first the following potential analysis of this case:

(78) Every $\mathbf{S}_z\ {}^{B}[_{\text{DP}}\ t^z$ one who brought any^y valuables]
$\mathbf{S}_z\ {}^{L}_{B}[_{\text{S}}\ {}_{B}t^x$ locked them$_y$ up]

As in the umbrella case above, this analysis is *compatible* with a scenario where each person locked up all of their (own) valuables, but does not *require* this. And yet, there seems to be a reading of (74) that does require this total locking up. We consider it likely that the best account will turn out to involve a variety of scalar implicature—when the speaker refers to a person z bringing a set of valuables v, the speaker is unlikely to have in mind a set v that is some undistinguished proper subset of the (entire) set of valuables that z brought.

Nonetheless, a semantic account that entails totality is available, and we present it for consideration. It is not implausible that, in addition to the treatment of *any* as a simple indefinite, the option of treating it as a generalized quantifier is available. For example, (79) translates as (80). The value of label B is $\ulcorner\textsc{valuables}([v]) \wedge \textsc{brought}(z,[v])\urcorner$. Thus, the summation pronoun [them $\mathbf{S}_v\ {}_B\langle\rangle$] denotes the set of valuables that z brought, and includes *all* valuables that z brought.

(79) Every $\mathbf{S}_z\ {}^{A}[_{\text{DP}}\ t^z$ one who$_z$ any $\mathbf{S}_v\ {}_{V}[t^v$ valuables$]$
$\mathbf{S}_v\ {}^{B}_{V}[t_z$ brought ${}_{V}t^v]]$
$\mathbf{S}_z\ {}^{L}_{A}[_{\text{S}}\ {}_{A}t^z$ locked [them $\mathbf{S}_v\ {}_{B}\langle\rangle$] up]

(80) $$\textsc{every}\left(\Sigma_z{}^{A}\textsc{some}\left(\begin{array}{c}\Sigma_{v}{}_{V}\textsc{valuables}([v]),\\ \Sigma_{v}{}^{B}_{V}\textsc{brought}(z,[v])\end{array}\right),\Sigma_z{}^{L}_{A}\textsc{locked}(z,\Sigma_{v}{}_{B}\mathbb{T})\right)$$

Strong Donkey Pronouns. The example we just considered is quite similar to the so-called strong reading of singular donkey pronouns, where the nuclear scope must be true of all instantiations of the donkey pronoun (e.g., x brought all x's umbrellas or locked up all x's valuables). And analyzing the "donkey" indefinite as a generalized quantifier derives precisely the exhaustive reading for cases like the following:

(81) Everyone who brought something valuable locked it up.

As with the umbrellas, (81) is again most acceptable in a situation where each person brought exactly one valuable item. And yet, hearers will accept the sentence when a few people brought more than one. In such a case, the most salient reading is the strong donkey reading, in which each person who brought multiple valuable items locked up all of their valuables. Again, the best analysis is probably pragmatic—the speaker presupposes a single valuable item, and the exhaustive reading arises when the hearer extrapolates to what the speaker *would* have said if they had considered the more general case—but a semantic analysis is also possible, as follows:

(82) Every $\mathbf{S}_z\ {}^{A}[_{\text{DP}}\ t^z$ one who$_z$ some $\mathbf{S}_v\ {}^{T}[_{\text{DP}}\ t^v$ thing valuable$]$
$\mathbf{S}_v\ {}^{B}_{T}[_{\text{S}}\ t_z$ brought ${}_{B}t^v]]$
$\mathbf{S}_z\ {}^{L}_{A}[_{\text{S}}\ {}_{A}t^z$ locked [it $\mathbf{S}_v\ {}_{B}\langle\rangle$] up]

The value of label B contains an external occurrence of z, and the summation pronoun [it $\mathbf{S}_v\ {}_{B}\langle\rangle$] denotes the (entire) set of valuable things that z brought. If no one brought more than one valuable item, that set is always a singleton, and the singular presupposition of *it* is satisfied. It seems, though, that this presupposition may be relaxed in edge cases, where v is singular for most values of z, even if it is plural for a a minority. (The singular presupposition of *something* must be relaxed similarly.)

Conditional Donkey Pronouns. While the system we present here is entirely extensional, we can analyze donkey anaphora in conditional sentences at least as well as other extensional systems, such as DPL or DRT. As is often done, we have taken the material conditional $p \to q$ to be shorthand for $\neg(p \wedge \neg q)$; see (14b) above. A "dynamic conditional" may be analogously defined, using "dynamic negation" (14d):

(83) if $\rightsquigarrow\ \lambda p \lambda q \,.\, {\sim}(p \wedge {\sim}q)$

This gives us (84b) as the FOL-PLUS translation for (84a); a FOL paraphrase is also provided. (Recall that $\exists\phi$ is equivalent to ϕ if ϕ has no local variables.)

(84) a. If any girl owned an umbrella, she brought it today.
b. $\neg\exists\left(\textsc{girl}([x]) \wedge \textsc{umb'lla}([u]) \wedge \textsc{owns}(x,u) \wedge \neg\exists(\textsc{bro't}(x,u))\right)$
$\equiv \neg\exists x \exists u(\textsc{girl}(x) \wedge \textsc{umb'lla}(u) \wedge \textsc{owns}(x,u) \wedge \neg\textsc{bro't}(x,u))$

This account is adequate within the limitations of an extensional system. A more sophisticated account would analyze conditionals as modal generalized quantifiers, parallel to the generalized quantifiers we discussed above, but a proper development would take us well beyond the scope of the current paper.

Discussion. The analysis of donkey anaphora presented here hinges on two key elements: (1) formula labels introduce a covert copy of the restrictor within the nuclear scope of a quantifier, and (2) an indefinite in the covert formula may be referenced in the overt expression. For example, in (67), the restrictor $O = \textsc{umbrella}([u]) \wedge \textsc{owns}([z],u)$ is included by label in the nuclear scope $O \wedge \textsc{brought}([z],u)$, permitting reference to the umbrella u. Restrictor and nuclear scope are separately closed by summation operators $\mathbf{S}_z \rightsquigarrow \Sigma_z$.

This approach is different both from standard E-type analyses, which treat donkey pronouns as hidden definites (e.g., *the umbrella x owns*), and from standard dynamic analyses, which directly extend the scope of the indefinite from the restrictor to the nuclear scope. Our hybrid approach side-steps many of the well-known problems with the two traditional approaches, described in Section 4, such as *The Problem of the Formal Link*, *The Proportion Problem*, and *The Problem of Indistinguishable Participants.*

2.8 Subordination

The phenomenon of quantificational subordination is illustrated in (85), with a first sentence, followed by two different possible second sentences:

(85) Almost every student brought an umbrella today.
a. Most (of them) used it, too.
b. Every one (of them) who used it stayed dry.

Previous works (e.g. Karttunen 1969; Sells 1985) have noted that an indefinite embedded under a quantifier, such as *an umbrella* under *most students* in (85), may serve as an antecedent to pronouns in the nuclear scope of later quantifiers over the same individuals, such as *most (of them)* in (85a). Such later quantifiers are said to be **subordinate** to the previous quantifier, hence the name of the phenomenon. We extend the usual set of examples by observing that such pronouns may appear not only in the nuclear scope of subordinate quantifiers, but also in their restrictors, as in *every one (of them) who used it* in (85b).

Quantificational subordination is accommodated under our approach by adding a subscript formula label to the subordinate quantifier to indicate that it is formally anaphoric to the main quantification. As before, this formula label also appears on the trace of the quantifier (D) and thus also on the restrictor. The structure is otherwise identical to the generalized quantifier structure that we have assumed to now.

(86) Almost every^z $\mathbf{S}_z$ $^S[t^z$ student] $\mathbf{S}_z$ $^B_S[_St^z$ brought an^u umbrella]
a. $_B$Most^z $\mathbf{S}_z$ $^M_B[_Bt^z$ (of them)] $\mathbf{S}_x$ $_M[_Mt^z$ used it$_u$, too]
b. $_B$Every^z $\mathbf{S}_z$ $^E_B[_Bt^z$ one who used it$_u$] $\mathbf{S}_z$ $_E[_Et^z$ stayed dry].

In this way, quantificational subordination is simply another example of block threading (section 1.4.3). In garden-variety generalized quantification, the quantificational DP (e.g., *almost every student*) bears the formula label that appears on the restrictor (here, S), the DP's trace (here, $_St^z$) shares the same label by (58a), and the trace, by (58b), passes the label on to the nuclear scope (^B_S[*brought an umbrella*]). By that syntactic mechanism, the restrictor meaning is threaded into the nuclear scope. What is special about quantificational subordination is that the subordinate quantifier bears an anaphoric label, and the mechanism just described thereby causes the antecedent block to be threaded into the meaning of the restrictor. An indefinite in any imported block may be referenced by a simple pronoun—thus u in the restriction of (86b) and in the nuclear scope of (86a).

Discussion. There appears to be an interesting connection between the uppercase and lowercase indices on a subordinate quantificational determiner. For instance, consider the two sentences below:[8]

(87) Most^z girls $^W[t^z$ wrote a^p final paper].
a. Each^p of [them $\mathbf{S}_p$ $_W\langle\rangle$] was a masterpiece.
b. #Each^p of [them $\mathbf{S}_p$ $_W\langle\rangle$] earned her$_z$ an A.

(87a) is a straightforward cases of a summation pronoun whose value derives from an indefinite in a previous sentence. Here, $\mathbf{S}$ sums up all the papers p written by a girl x, and *each (of)* distributes over this sum individual. This is not a case of subordination, and does not require an uppercase variable subscript W on the quantifier *each*. Notice that the block introduced by the $\mathbf{S}$ in the summation pronoun does not extend beyond the summation pronoun itself.

[8]Lecture notes by Heim credit Keny Chastain with a similar observation.

Thus no occurrence of z outside of this block can refer to the girls mentioned in W. And indeed, the attempted reference to a girl in (87b) sounds quite odd.

But what if we choose to interpret *each* as subordinate:

(88) $\#_W\text{Each}^p\ {}^{E}_{W}[_W t^p$ (of them)$]\ {}_E[_E t^p$ earned her$_z$ an A].

Here the restriction, subscripted W, is anaphoric to the previous nuclear scope block W. Since E resumes the thread of W, and W contains the indefinite z, it should be possible to refer to z (the girl that wrote the paper) in the nuclear scope, but the sentence sounds quite odd.

Subordination is in fact not freely available. The conditions that permit it are not entirely well-understood, but for this particular case we can give a reasonable syntactic explanation. Consider again the generalized quantifier structure (63). Recall, in particular, that the root node, by (58b), bears the lowercase index of the head quantifier and the uppercase label of the nuclear scope. Thus, in (87), the root sentence has labels ${}_W\text{S}^z$, with z from the head *most*^{z} and W from the nuclear scope. But the anaphoric quantifier in (88) has labels ${}_W\text{each}^p$. If an anaphoric quantifier is subject to the same agreement condition (58a) that holds between a trace and its antecedent, the ill-formedness of (88) is explained as an index mismatch between anaphor and antecedent. This is at least one case in which subordination is not available. (In the well-formed cases of (86), the anaphoric quantifier agrees with its antecedent with respect to both indices.)

2.9 Conclusion

Our goals in this section have been twofold: (1) to show that the empirical coverage of FOL-PLUS includes the full gamut of cases of improper binding that have been discussed in the literature, and (2) to provide an apples-to-apples comparison between FOL-PLUS and E-type approaches, showing that the FOL-PLUS account is more explicit and satisfactory than E-type accounts at several points. In the dynamic semantics literature, logical translations of natural-language sentences are often given on the basis of intuition, rather than formally computed by a translation function. We have explicitly presented the FOL-PLUS translations of many of the examples, and we hope those translations are sufficient to convince the reader that it is fair to compare our translations to the intuitional translations of the dynamic literature.

3 FOL-PLUS and dynamic logic

3.1 Precis

In this section, we consider the relationship between static logics and dynamic logics. As representative of dynamic logics, we take Dynamic Predicate Logic (DPL) (Groenendijk and Stokhof 1991), which provides a foundation for systems such as those of van den Berg (1996) and Brasoveanu (2007), and is formally essentially identical to Discourse Representation Theory (DRT) (Kamp 1981); see van Eijck (1999) for a formalization of DRT that brings out the similarities to DPL. By *dynamic logic,* we understand a system that defines the meaning $[\![\pi]\!]$ of an expression π to be a nondeterministic mapping from some input state to an output state, and we understand *static logic* to be a system in which $[\![\phi]\!]^g$ is a truth value, or, equivalently, in which $[\![\phi]\!]$ is a set of assignments.

Groenendijk and Stokhof themselves compare their dynamic logic DPL to FOL (the canonical static logic), concluding that DPL is strictly more expressive than FOL. Their argument implicitly rests, however, on the assumption that the proper basis for such a comparison is via truth conditions, defining an expression of dynamic logic (henceforth, a **program**) to be true just in case it produces an output. Under this assumption, dynamic meanings are clearly finer-grained than static meanings, inasmuch as pairs of programs exist that have the same truth conditions but different input-output mappings, whereas it is impossible for two programs to compute the same mapping but to differ in truth conditions. This view has been taken up by much of the later literature, as well. For example, the *Stanford Encyclopedia of Philosophy* asserts that "what dynamic semantics provides is a generalization of truth conditional semantics .... The classical meanings become the *preconditions* for success of the discourse actions. ... classical meanings are recoverable from the relational dynamic meanings as *projections* onto their 'input' coordinate" (Nouwen et al. 2022, their emphasis).

We do not dispute that truth conditions are coarser-grained than input-output mappings, but we do dispute the idea that truth conditions are the proper basis for comparison. The asymmetry that Groenendijk and Stokhof observe is not intrinsic to static logics, but to truth conditions. A static formula *also* defines an input-output mapping, in a natural and familiar way, and when we use input-output mappings for comparison, the assymetry disappears. In particular, we will show that the FOL subset of FOL-PLUS is strongly equivalent to DPL, in the sense that for every program there is a corresponding formula that defines the same input-output mapping, and vice versa.

Formulas are widely used in mathematics to define mappings. For example, one typically represents the unit circle by the equation $x^2 + y^2 = 1$, understanding it to implicitly define the mapping ("multi-valued function") $y = \pm\sqrt{1 - x^2}$. The implicit representation is used for example in the technique of implicit differentiation, where one differentiates the function by applying differentiation rules to the equation that implicitly represents it. More broadly, mathematics texts are rife with expressions like "consider y as a function of x in the equation ϕ." Even beginning algebra students are familiar with this notion, when they are asked to "solve for y" in an equation involving variables x and y. And in addition to characterizing the structures that satisfy it, a fundamental use of a model is to define relations within a given structure: a relation is **first-order definable** just in case (glossing over some technical details) there is a formula of FOL that implicitly defines it; see for example Boolos and Jeffrey (1980).[9]

[9]Using open formulas to define relations, and viewing the satisfaction set $[\![\phi]\!]$ as the graph of a relation, are fundamental to **relational algebra** in the theory of databases (Codd 1970;

Consider an open formula of FOL, such as:

(89) $\textsc{p}(x) \wedge \textsc{q}(x, y)$

Call that formula ϕ. It determines a particular relation between its free variables. Thinking of the relation as a nondeterministic mapping in which x is the independent variable (the input) and y is the dependent variable (the output), we write the relation as $\boldsymbol{f}_\phi$:

(90) $\boldsymbol{f}_\phi(x, y)$ iff $\textsc{p}(x) \wedge \textsc{q}(x, y)$

More generally, there may be multiple independent variables $\boldsymbol{x} = (x_1, \ldots, x_m)$ and multiple dependent variables $\boldsymbol{y} = (y_1, \ldots, y_n)$. In the general case, we have:

(91) $\boldsymbol{f}_\phi(\boldsymbol{a}, \boldsymbol{b})$ iff $\exists h(h(\boldsymbol{x}) = \boldsymbol{a} \wedge h(\boldsymbol{y}) = \boldsymbol{b} \wedge [\![\phi]\!]^h = 1)$,

where $h(\boldsymbol{x})$ abbreviates $h(x_1), \ldots, h(x_m)$. Conversely:

(92) $[\![\phi]\!]^h = 1$ iff $\boldsymbol{f}_\phi(h(\boldsymbol{x}), h(\boldsymbol{y}))$.

To use a formula to define a mapping, via (91), we require one thing beyond the formula itself, namely, a way of distinguishing between dependent and independent variables. FOL-PLUS has a ready-made mechanism for this purpose: the brackets that mark local variables. Let us define the above-mentioned "FOL subset of FOL-PLUS" to be the language that remains if we exclude summation and formula labels, and if we restrict the domain to singleton pluralities (individuals); let us call the resulting language FOL-U. FOL-U differs from FOL only in replacing $\exists x(\ldots)$ with $\exists(\ldots[x]\ldots)$. Moreover, we understand the brackets to mark dependent variables. That is, the dependent variables are the local variables and the independent variables are the external variables: $\boldsymbol{y} = L_\phi$ and $\boldsymbol{x} = X_\phi$, where X_ϕ is the set of free variables of ϕ that are not in L_ϕ.

Turning now to dynamic semantics, a program defines a mapping from machine state to machine state, where Groenendijk and Stokhof took a machine state to be exhaustively characterized by the values it assigns to variables in memory. As a technical expedient, they used assignments to represent states, an assignment being a total function whose domain is the (countably infinite) set of variables of the language, and they took the **dynamic relation** defined by a program to be a relation $\boldsymbol{D}(g, h)$ between pairs of assignments g and h. But more basic is the **input-output mapping** $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y})$, where, in the case of a program, $\boldsymbol{x} = (x_1, \ldots, x_m)$ is the set of *input variables* and $\boldsymbol{y} = (y_1, \ldots, y_n)$ is the set of *output variables.* The dynamic relation is then:

(93) $\boldsymbol{D}(g, h)$ iff $\boldsymbol{f}(g(\boldsymbol{x}), h(\boldsymbol{y}))$. [revised below]

Now Groenendijk and Stokhof impose an additional constraint on the relationship between input state g and output state h: they require not only that g

Ullman 1998). Relational algebra does not, however, distinguish between dependent and independent variables—or rather, a choice of which variables to include in the output is made on a query-by-query basis.

and h correctly represent the mapping from input values to output values, but also that g and h differ only with respect to output variables. That is, instead of (93), they adopt:

(94) $\boldsymbol{D}(g,h)$ iff $\boldsymbol{f}(g(\boldsymbol{x}),h(\boldsymbol{y})) \wedge g[\boldsymbol{y}]h$.

This additional constraint seems undesirable. Intuitively, an implementation that maps the values of $\boldsymbol{x}$ to values of $\boldsymbol{y}$ correctly, but also changes the contents of unused memory cells, does not thereby become an incorrect implementation, and yet (94) rules it out. Be that as it may, we accept (94) as established practice.

For a DPL program π, the dynamic relation is the meaning $[\![\pi]\!]$:

(95) $\boldsymbol{D}_\pi = [\![\pi]\!]$. [Def.]

where $[\![\pi]\!]$ is defined in Figure 3.

(96) a. $[\![P(\alpha_1,\ldots,\alpha_n)]\!](g,h)$
iff $g = h \wedge [\![P(\alpha_1,\ldots,\alpha_n)]\!]^h = 1$ [pred. appl.]
b. $[\![\alpha = \beta]\!](g,h)$ iff $g = h \wedge [\![\alpha = \beta]\!]^h = 1$ [equation]
c. $[\![\,[\alpha]\,]\!](g,h)$ iff $g\backslash\alpha = h\backslash\alpha$ [assignment]
d. $[\![\sim\rho]\!](g,h)$ iff $g = h \wedge \neg\exists k([\![\rho]\!](g,k))$ [dyn. negation]
e. $[\![\rho_1;\rho_2]\!](g,h)$ iff $\exists k([\![\rho_1]\!](g,k) \wedge [\![\rho_2]\!](k,h))$ [dyn. conjunction]

Figure 3: Formal definition of the interpretation function $[\![\pi]\!]$ of Dynamic Predicate Logic (DPL).

For a FOL-U formula ϕ, substituting (92) into (94) gives us a natural, independent definition of the dynamic relation defined by a static formula:

(97) $\boldsymbol{D}_\phi(g,h)$ iff $[\![\phi]\!]^h = 1 \wedge g[L_\phi]h$. [Def.]

Our main goal is to show that FOL-U is strongly equivalent to DPL, in the sense that there is a one-one correspondence between DPL programs π and FOL-U formulas π' such that:

(98) $\boldsymbol{D}_{\pi'} = \boldsymbol{D}_\pi$.

Once we have established a strong equivalence between FOL-U and DPL, it becomes trivial to introduce non-singleton pluralities, summation (Σ_x), and formula labels into DPL. The result can be thought of as a dynamic "view" of FOL-PLUS. It is directly comparable to, but vastly simpler than, dynamic systems such as those of van den Berg or Brasoveanu, and yet, as we have seen in the previous section, it covers all significant improper binding phenomena.

3.2 The translation from DPL to FOL-U

In order to prove (98), we must first establish a one-one correspondence between DPL programs and FOL-U. We adopt the translation function that is defined in Figure 4.

(99) a. $P(x_1, \ldots, x_n) \leftrightarrow P(x_1, \ldots, x_n)$.
b. $x_1 = x_2 \leftrightarrow x_1 = x_2$.
c. $[x] \leftrightarrow \mathbb{T}([x])$.
d. $\pi_1 ; \pi_2 \leftrightarrow \pi_1' \wedge \pi_2'$.
e. $\sim\pi \leftrightarrow \neg\exists\pi'$.

Figure 4: The FOL-U translation function $\pi' = \text{FOL-U}(\pi)$. The input π is a DPL program and the output π' is a formula of FOL-U.

Not every FOL-U formula is in the range of the FOL-U translation function, but it is easy to see that the translation function does constitute a one-one correspondence between DPL and a subset of FOL-U: one can reverse the operator substitutions of (99) to map π' back (unambiguously) to π.

The fact that the range of translation is a proper subset of FOL-U means that there are dynamic relations that are definable in FOL-U but not in DPL. An example is:

(100) $\neg P([x])$.

The dynamic relation $\boldsymbol{D}_{\neg P([x])}(g, h)$ holds just in case $\neg P(h(x)) \wedge g[x]h$, a dynamic relation that is not expressible in DPL.[10]

The FOL-U translation differs in an important way from the correspondence that Groenendijk and Stokhof implicitly assume when they compare DPL and FOL, which is given in Figure 5.[11]

(101) a. $P(x_1, \ldots, x_n) \rightsquigarrow P(x_1, \ldots, x_n)$.
b. $x_1 = x_2 \rightsquigarrow x_1 = x_2$.
c. $[x]; \pi \rightsquigarrow \exists x \pi'$.
d. $\sim\pi \rightsquigarrow \neg\pi'$.
e. $\pi_1 ; \pi_2 \rightsquigarrow \pi_1' \wedge \pi_2'$ [if (c) does not apply].

Figure 5: The Groenendijk and Stokhof Translation from DPL to FOL, $\pi' = \text{FOL-GS}(\pi)$.

[10]Some readers may consider that lack of expressiveness to be a positive thing, in that it "makes an empirical claim" about what is available in natural language. We see no advantage that such an "implicit claim" has over making an explicit assertion ("natural language negation always embeds an existential closure"), but one could certainly make the same "implicit claim" by adopting a version of FOL-U that uses $\sim$ in place of $\neg$.

[11]Groenendijk and Stokhof actually use FOL syntax for DPL; the FOL-GS translation function actually represents the conversion from the syntax of Figure 3 to the syntax used by Groenendijk and Stokhof.

Note that an existential closure $\ulcorner \exists x \urcorner$ occurs immediately in the translation of a bracketed variable $\ulcorner [x] \urcorner$. This contrasts with our translation FOL-U(π), where existential closure is delayed until the application of negation. For example:

(102) a. $[x]; \textsc{dog}(x) \overset{\text{FOL-U}}{\rightsquigarrow} \mathbb{T}([x]) \wedge \textsc{dog}(x)$
b. $[x]; \textsc{dog}(x) \overset{\text{FOL-GS}}{\rightsquigarrow} \exists x\, \textsc{dog}(x)$

As a consequence, under the FOL-GS translation, no bracketed variable ever occurs as a free variable, and it is impossible to use the FOL-GS translations as definitions of input-output mappings.[12]

Let us write $g \models \pi$ to mean that π is true under assignment g. Groenendijk and Stokhof define the truth conditions of a program π as:

(103) $g \models \pi$ iff $\exists h(\llbracket \pi \rrbracket (g, h))$.

For a formula ϕ, the truth conditions are:

(104) $g \models \phi$ iff $\llbracket \phi \rrbracket^g = 1$.

Groenendijk and Stokhof show that, where $\pi' = $ FOL-GS(π):

(105) $g \models \pi$ iff $g \models \pi'$.

The same is not true under the FOL-U translation. Rather, if $\pi' = $ FOL-U(π):

(106) $g \models \pi$ iff $g \models \exists \pi'$.

The $\ulcorner \exists \urcorner$ in (106) discards the information about the values of local variables in π', and in that way represents an obvious loss of information; that loss of information is what gives rise to the assymetry between truth conditions and dynamic relations that Groenendijk and Stokhof observe.

3.3 Proof of equivalence

Input and output variables. A first question is how to identify the input variables and output variables of a program. Identifying them will be facilitated if we view dynamic conjunctions as flat n-place conjunctions, in which each conjunct is is either a bracketed variable or a test (the tests being predicate applications, equations, and negations). Replacing binary conjunction with flat n-place conjunction is justified by the associativity of dynamic conjunction.

Let us begin with output variables. An output variable is one whose value in the output is set by the program, not simply copied from the input. In a DPL program, a statement of form $[x]$ sets the value of variable x. But not every occurrence of $[x]$ in a program sets a value that survives into the program output. In particular, if $[x]$ is encapsulated in a negation, the value that is

[12]The statement in the text assumes that the original program is in standard form (see section 3.3 below for the definition of standard form for programs). In non-standard examples like $P(x); [x]; Q(x)$, the x of $P(x)$ and the x of $Q(x)$ are (intuitively speaking) different variables with the same name.

assigned to x is essentially discarded: if $\sim\pi$ has an output on input g, that output is g itself.

Define the **blocks** of a program π to be π itself (the root block) and the complements of negations occurring anywhere in π (embedded blocks). A bracketed variable occurrence in an embedded block is "encapsulated" and does not affect program output. Conversely, a bracketed variable occurrence that is not in an embedded block is an output variable occurrence.

(107) x is an output variable of π just in case there is an occurrence of $[x]$ in π that is not contained in an embedded block.

An input variable is one whose value in the input is read by the program. Obviously, an input variable must occur in unbracketed form in the program, but that is not a sufficient condition. If x is the unbracketed occurrence, and a previous occurrence $[x]$ sets the value that x reads, then x obviously does not read the input value. We can define the input variables of program π recursively as follows:

(108)
a. If π is a bracketed-variable statement $[x]$, it has no input variables.
b. If π is of form $P(x_1, \ldots, x_n)$ or $x_1 = x_2$ $(n = 2)$, its input variables are $x_1, \ldots, x_n$.
c. If π is a negation of form $\sim\rho$, its input variables are the input variables of ρ.
d. If π is a conjunction $\rho_1; \ldots; \rho_n$, then x is an input variable of π iff x is an input variable of ρ_j and no ρ_i with $i < j$ has form $[x]$.

Standard form. It is possible for a variable to be both an input variable and an output variable, but only if the program **resets** its value.

(109) The variable x is reset in π just in case π is a conjunction $\rho_1; \ldots; \rho_n$, x is an input variable of ρ_i, and ρ_j has form $[x]$, for $i < j$.

Let us say that a block is in **standard form** iff it does not reset any variables. A program is in standard form iff all of its blocks are in standard form. If a program is in standard form, we can drop the condition "$i < j$" in (108d), and the sets of input variables and output variables are disjoint.

In general, dynamic conjunction is not commutative, because reordering conjuncts can change binding relationships, as illustrated in the difference between $P(x); [x]$ and $[x]; P(x)$. However, for an expression that is in standard form, certain reorderings are meaning-preserving. In particular, if a conjunction $\pi_1; [x]$ is in standard form, then x is not an input variable of π_1, hence if $g[x]h$ then $[\![\pi_1]\!](g, g)$ is true iff $[\![\pi_1]\!](h, h)$. Thus $\pi_1; [x]$ is equivalent to $[x]; \pi_1$. That is, in standard form, bracketed variables may be moved to the front of a conjunction without changing the dynamic meaning.[13] We may then without loss of generality assume that all conjunctions have the form:

[13] Even in standard form, dynamic conjunction is not fully commutative; we may not reorder $[x]; P(x)$ to $P(x); [x]$. But reorderings in which bracketed variables move leftward are valid.

(110) $[x_1]; \ldots; [x_m]; \pi_1; \ldots; \pi_n$

where $x_1, \ldots, x_m$ are the output variables of the conjunction and each π_i is a test (namely, a predicate application, equation, or negation).

With that rearrangement, it becomes straightforward to show that, for any DPL program π and FOL-U translation π':

(111) a. The input variables of π are $X_{\pi'}$.
b. The output variables of π are $L_{\pi'}$.
c. π is in standard form iff π' is in standard form.

We require one additional lemma: tests have no output variables. By definition, an output variable is one whose value may differ between input and output, but for a test π (by definition), $[\![\pi]\!](g, h)$ iff $g = h$ and $[\![\pi]\!](g, g)$.

Main proof. We turn now to the equivalence of DPL and FOL-U (98). We will prove equivalence only under the assumption of standard form. We consider that to be a mild assumption, however, on the grounds that (1) we assumed standard form exclusively in the empirical discussion of section 2, (2) we are unaware of analyses in the literature that use $[x]$ to reset the value of a variable whose value has previously been set, and (3) a program π that is not in standard form can be converted to standard form by renaming output variables that appear as input variables. Renaming output variables does change the dynamic relation $\boldsymbol{D}_\pi$ but it does not change the input-output mapping $\boldsymbol{f}_\pi$, and we consider the input-output mapping to be definitive.

We wish to show that, for any DPL program π in standard form, with $\pi' = \text{FOL-U}(\pi)$, we have $\boldsymbol{D}_{\pi'} = \boldsymbol{D}_\pi$. That is, substituting in definitions (95) and (97), we wish to show that, for all g and h:

(112) $[\![\pi]\!](g, h)$ iff $g[L_{\pi'}]h \wedge h \in [\![\pi']\!]$.

The proof is recursive on the structure of the program. The base cases are the atomic formulas. Recall that $[\![\pi]\!]$ is defined in Figure 3 and $[\![\pi']\!]$ is defined in Figure 2 (omitting the clauses for Σ_x and formula labels), and we make frequent implicit use of the clauses of those definitions.

(113) a. For π of form $\ulcorner P(x_1, \ldots, x_n)\urcorner$ or $\ulcorner x_1 = x_2\urcorner$ $(n = 2)$, we have $[\![\pi]\!](g, h)$ iff $g = h$ and $h \in [\![\pi']\!]$, which (because $L_{\pi'} = \emptyset$) is equivalent to: $[\![\pi]\!](g, h)$ iff $g[L_{\pi'}]h \wedge h \in [\![\pi']\!]$.
b. For π of form $\ulcorner[x]\urcorner$, we have $[\![\pi]\!](g, h)$ iff $g[x]h$, $L_{\pi'} = \{x\}$, and $h \in [\![\pi']\!]$ always. Thus $[\![\pi]\!](g, h)$ iff $g[L_{\pi'}]h \wedge h \in [\![\pi']\!]$.

The recursive cases are conjunction and negation. By recursive hypothesis, $[\![\rho]\!](g, h)$ iff $g[L_{\rho'}]h \wedge h \in [\![\rho']\!]$, for all subexpressions ρ.

(114) a. If π is a conjunction, it may be written in form $[x_1]; \ldots; [x_m]; \rho_1; \ldots; \rho_n$, where all ρ_i are tests. The FOL-U translation has form $\mathbb{T}([x_1]) \wedge \ldots \wedge \mathbb{T}([x_m]) \wedge \psi_1 \wedge \ldots \wedge \psi_n$. Since the ρ_i are tests, we have $L_{\rho_i'} = \emptyset$,

and $[\![\rho_i]\!](g,h)$ iff $g = h \wedge [\![\rho_i]\!](h,h)$, iff $g = h \wedge h \in [\![\rho_i']\!]$. Thus $[\![\pi]\!](g,h)$ iff $g[x_1,\ldots,x_m]h$ and $h \in [\![\rho_1' \wedge \ldots \wedge \rho_n']\!]$. The conjuncts $\mathbb{T}([x_i])$ are tautologically true, so $h \in [\![\rho_1' \wedge \ldots \wedge \rho_n']\!]$ iff $h \in [\![\pi']\!]$. Since $L_{\rho_i'} = \emptyset$ for all ρ_i, the local variables of π' are $x_1,\ldots,x_m$, yielding $[\![\pi]\!](g,h)$ iff $g[L_{\pi'}]h \wedge h \in [\![\pi']\!]$.

b. If π is a negation, it has form $\sim\!\rho$, and $[\![\pi]\!](g,h)$ iff $g = h \wedge [\![\pi]\!](g,g)$ iff $g = h \wedge \neg\exists k([\![\rho]\!](g,k))$. Using the recursive hypothesis, the latter is equivalent to $\neg\exists k(g[L_{\rho'}]k \wedge k \in [\![\rho']\!])$. Since π' is the FOL-U translation of π, it has the form $\neg\exists\rho'$; thus, by (11e) and (3c), $h \in [\![\pi']\!]$ iff $\neg\exists k(g[L_{\rho'}]k \wedge k \in [\![\rho']\!])$. Thus $[\![\pi]\!](g,h)$ iff $g = h \wedge h \in [\![\pi']\!]$, iff $g[L_{\pi'}]h \wedge h \in [\![\pi']\!]$ (since $L_{\pi'} = \emptyset$). ■

4 Assessing the Two Major Approaches

4.1 E-type Approach

Classic E-type proposals assume "free variables [which] refer to contextually salient entities of the appropriate type," (Heim 1990, p. 139) such as a salient function mapping a farmer to a donkey. However, they are vague about what qualifies as contextually salient. This leads to overgeneration problems, as illustrated in (115), due to Heim (1990). Even though the successor function (mapping each integer to the next higher one) is made quite salient, it cannot support *it* as an E-type pronoun. This example illustrates the so-called *Problem of the Formal Link* (Kadmon 1987), the observation that even E-type pronouns require a nominal antecedent. In this same vein, Heim (1982) points out cases like (116), where two phrases with near identical meanings nevertheless have different anaphoric potential. Early E-type analyses could not explain how the function mapping men to wives would be salient in (116a) but not (116b).

(115) #Speaking of the successor function, every number is smaller than it.
(Intended: *Every number is smaller than its successor.*)

(116) a. Every man who has a wife sat next to her.
b. ??Every married man sat next to her.

The E-type approach also suffers from undergeneration, arising from the uniqueness that singular E-type pronouns require to make the extension of their description a single individual. Again, Heim (1982) provides the crucial example: although *it* in (117) must invoke a salient function mapping people to the sage plants they bought, the sentence itself asserts that this function never has a unique output since each person bought, say, a flat of nine plants.

(117) Everybody who bought a sage plant bought eight others along with it.

Elbourne (2005) undertakes to solve these problems. To stem the overgeneration problem, he proposes that E-type pronouns involve NP deletion, a form of ellipsis targeting a recently mentioned NP. For instance, under his analysis, the pronoun *her* in (116a) has the same denotation as the definite description *the wife*, with the NP *wife* deleted based on its mention earlier in the sentence. However, *her* in (116b) lacks any such NP to support a similar ellipsis.

The undergeneration problem in (117) requires a bit more machinery, though, since the pronoun here would essentially mean *the sage plant* under Elbourne's analysis, and this scenario necessarily includes many multiples of nine plants. Following Heim (1990), Elbourne solves this problem by increasing the complexity of the semantic system. Determiners like *every* are no longer simple quantifiers over individuals; rather, they operate over individuals, minimal situations containing these individuals (for the restriction), and extensions of these minimal situations (for the nuclear scope). For instance, a simplified version of Elbourne's rendering of (117) is given in (118):

(118) For every x and every minimal situation s such that x bought a sage plant in s, there is a situation s' extending s such that x bought eight other sage plants in s' along with the (unique) sage plant in s.

Crucially, while the nuclear scope is evaluated with respect to an extended situation s', the pronoun *it* itself must be evaluated in the minimal situation s satisfying the restriction.

This solution is not quite enough to solve another class of counterexamples, dubbed *The Problem of Indistinguishable Participants* by Heim (1990) and attributed therein to Hans Kamp, illustrated in (119):

(119) If a bishop meets another bishop on the road, he blesses him.

The simple "minimal situation restriction" move mentioned above will not suffice here, since the restriction itself mentions two bishops without significantly distinguishing them. Instead, therefore, Elbourne multiplies the number of situations involved (up to seven for (119)).[14,15] He then "distinguishes" one of the bishops by referencing the number of situations they appear in overall, but this procedure is not precisely formalized.[16]

[14]Here are his truth conditions for (119):

(i) λs_5 .
 for every minimal $s_6 \leq s_5$ such that $\exists$ an x and s_1 such that:
  s_1 is a min. situation s.t. $s_1 \leq s_6$ and x is a bishop in s_1, s.t. $\exists$ an $s_2 \leq s_6$ s.t.:
   s_2 is a min. situation s.t. $s_1 \leq s_2$ and there is a y and s_3 s.t.:
    s_3 is a min. situation s.t. $s_3 \leq s_2$ and y is a bishop in s_3, s.t. $\exists$ an $s_4 \leq s_2$ s.t.:
     s_4 is a min. situation s.t. $s_3 \leq s_4$ and x meets y in s_4;
 there is an $s_7 \leq s_5$ such that:
  s_7 is a min. situation s.t. $s_6 \leq s_7$ and the bishop in s_7 **"distinguished" by being in fewer situations overall** blesses (in s_7) the **other** bishop in s_7.

[15]In later work, Elbourne (2016) revisits this analysis in view of a separate proposal involving definite descriptions. He essentially maintains the same analysis, but is forced to make the truth conditions even slightly more complex in order to accommodate the new proposal.

[16]Elbourne also points out that more symmetrical cases like (i) sound odd, which he claims

One final issue involving the uniqueness of E-type accounts, also described recently in Mandelkern and Rothschild (2020) and Lewis (2012), involves simple cross-sentential cases:

(120) A girl was eating lunch in a crowded cafeteria, surrounded by a sea of other girls. She got up and left, leaving a single purple ribbon behind.

Here, there is no unique referent fitting a description as required by the use of *she* in the second sentence: *the girl*, *the girl eating lunch surrounded by other girls*, etc., all describe multiple individuals. Presumably, an E-type account could make the same move as Elbourne, positing a raft of minimal situations generated by the first sentence and referred back to by the second sentence, but we have not seen this sort of analysis mooted for cross-sentental cases like this.

4.2 Dynamic Approach

The dynamic approach essentially side-steps all the issues above by allowing indefinites to extend their scope beyond the traditional limits: matrix-level indefinites effectively scope at the discourse level, allowing unlimited cross-sentential anaphora without requiring uniqueness or unformalized salient descriptions in the context. Apparently indistinguishable participants can therefore be distinguished using the usual methods of indexing and binding.

This approach comes with its own complexities, though, including a shift from propositional denotations to relational ones, pairing input and output states. (See below for discussion of the non-determinism inherent to such relational meanings.) And like the E-type approach, the complexities of the dynamic approach multiply as it attempts to cover more empirical ground.

The early dynamic systems used relations over input and output states consisting of single assignments (Groenendijk and Stokhof 1991), or semi-equivalently used functions over states comprising sets of assignments (Kamp 1981; Heim 1982). For instance, as shown above, under Dynamic Predicate Logic (Groenendijk and Stokhof 1991), a simple sentence like (121a), with no pronouns, takes any single assignment as its input; and its possible output states include all assignments g such that $g(f)$ is a farmer and $g(d)$ is a donkey that $g(f)$ owns. A simple sentence like (121b), with pronouns but no indefinites, restricts its input assignments g to those where $g(f)$ beats $g(d)$; and its output state is identical to its input.

(121) a. A^f farmer owns a^d donkey. b. He$_f$ beats it$_d$.

as a prediction of his account. However, there seems to be more going on here, since a case like (ii) also sounds odd, even though the participants are distinguishable (see also Barker and Shan 2008; Elbourne 2009):

(i) #If a bishop and a bishop meet on the road, he blesses him.

(ii) #If a dog and a cat fight in the road, it bites it.

But these early systems did not handle plural phenomena like quantificational subordination, and they even had trouble with generalized quantifiers like *most*. For instance, *every* in DPL asserts that every output state of its restriction is a possible input state for its nuclear scope. For (122), this requires that every assignment g pairing a farmer $g(f)$ with a donkey $g(d)$ that $g(f)$ owns be such that $g(f)$ beats $g(d)$. But a simple extension of this analysis to (123) does not work. Sentence (123) does not mean that most pairings of a farmer with a donkey that the farmer owns are such that the farmer beats the donkey; if it did, a single malicious farmer who owns and beats a hundred donkeys could verify the sentence even if fifty gentle farmers each owned a single donkey and treated it well. Kadmon (1987) dubbed this *The Proportion Problem*.

(122) Every^f farmer who owns a^d donkey beats it$_d$.

(123) Most^f farmers who own a^d donkey beat it$_d$.

To solve both these issues, van den Berg (1996) moves from relations over states comprising single assignments to states comprising sets of assignments. This complexification of data structures requires a commensurate complexification of interpretative rules, and of the logical formulas assigned to sentences. For instance, random assignment in a van den Berg-style system requires checking both that every assignment in the input state relates to at least one assignment in the output state and that every assignment in the output state relates to at least one assignment in the input state. For each input state, the set of outputs consists of all states that pass the check. With complex context states in hand, constructing an account of generalized quantifiers and quantificational subordination itself requires additional complexities: a dummy individual $\star$ to simulate missing values; a special "structured inclusion" operator relating input states to subsets thereof based on precise rules involving dummy individuals; and special maximality and distributive operators also made complex by the need to accommodate dummy individuals. The full formal details for such a system, including definitions for around a dozen special operators, can be found in Brasoveanu (2007) starting on pp. 195–199 and finishing on pp. 255–260.

The empirical issues with dynamic accounts mostly involve technical problems in the strategies required to extend the scope of indefinites. For instance, in order to capture cases like (124a) (similar to (49) above), dynamic accounts usually cap indefinite scope within negation. And yet, as (124b) (similar to (48)) shows, indefinites sometimes do scope out of (especially double) negation:

(124) a. John doesn't own a car. #It's (just) in the shop.
b. It's not like John doesn't own a car. It's (just) in the shop.

Another case that dynamic approaches cannot handle easily is paycheck pronouns (Karttunen 1969). As mentioned above, the dynamic approach essentially involves extended scope binding from an antecedent to a bound pronoun. However in cases like (125), the apparent antecedent (*her paycheck*) does not denote the same individual as the later pronoun *it*; no simple binding structure could achieve the correct reading.

(125) The employee who saved her paycheck was wiser than the one who spent it.

Finally, plural dynamic systems following van den Berg suffer from an empirical deficit involving reference in the nuclear scope to a summation plural whose antecedent is in the restriction of the same quantifier (Nouwen 2003b). For instance, *them* in (126) may refer to all the donated toys, summing over the people who donated them within a quantification over these same people. In order to solve this problem, Nouwen (2003b) increases the complexity again, introducing assignments whose values are stacks of individuals rather than single individuals or sets of individuals.

(126) Everyone who donated a toy helped wrap them all for the charity drive.

4.3 DUAL

Keshet (2018) presents Dynamic Update Anaphora Logic, a dynamic system that takes a very different approach from van den Berg-style systems, such as Brasoveanu (2007). Essentially, DUAL is a version of DPL with two additions: labels for subformulas and complex terms collecting all values for a term in such a stored subformula. These latter serve the same purpose as the summation terms of FOL-PLUS, giving translations like the following for quantified sentences:

(127) Most^d dogs^D bark^B

$$\rightsquigarrow (D : \exists d\, \textsc{dog}(d)) \wedge (B : D \wedge \textsc{bark}(d)) \qquad\qquad \wedge\, \textsc{most}(D.d, B.d)$$
$$\rightsquigarrow (D : \exists d\, \textsc{dog}(d)) \wedge (B : \exists d\, \textsc{dog}(d) \wedge \textsc{bark}(d)) \wedge \textsc{most}(D.d, B.d)$$

The clauses prefixed with a capital letter variable are vacuously true; their only contribution is to store a full DPL denotation in the capital letter variable. This denotation may be retrieved by later use of the same variable, either as a clause on its own, or in complex terms. The two complex terms $\ulcorner D.d \urcorner$ and $\ulcorner B.d \urcorner$ are equivalent to the summation terms $\ulcorner \Sigma_{d\ D}\mathbb{T} \urcorner$ and $\ulcorner \Sigma_{d\ B}\mathbb{T} \urcorner$ in FOL-PLUS, returning all dogs and all dogs that bark, respectively.

We very much view FOL-PLUS as an intellectual successor to DUAL, especially in its use of labeled subformulas, the main innovation of DUAL (equivalents to complex/summation terms were introduce by Kamp and Reyle (1993)). In fact there are very few differences in empirical coverage when you compare FOL-PLUS to DUAL, although we point out one:

Plurals Brasoveanu (2008) argues at length for two kinds of plurals in natural language, roughly corresponding to the denotations of plural indefinites versus plural summation pronouns. DUAL only includes this latter type of plural, since it follows DPL in only allowing singular individuals as the denotations of (lowercase) variables. All plurals in DUAL are constructed via complex terms, even those introduced by plural indefinites (see Keshet 2018, p.290 and footnotes 12 and 17). And since complex terms in DUAL are necessarily maximal, issues arise similar to the E-type treatment of cross-sentential anaphora. Consider the following plural version of (120) above:

(128) a. Two^g girls^G [were eating lunch in a crowded cafeteria]^E_G, surrounded by a sea of other girls.
b. They$_{E.g}$ got up and left.

The label E in (128) will denote the formula translating (roughly) "a^g girl was eating lunch in a crowded cafeteria," and thus $\ulcorner E.g \urcorner$ will necessarily denote <u>all</u> girls eating in the cafeteria. Two problems arise, therefore: first, (128a) will come out as false, since $\ulcorner E.g \urcorner$ denotes more than two girls. Next, the pronoun *they*$_{E.g}$ denotes every girl there, rather than the two mentioned in the first sentence.

Although the empirical differences between DUAL and FOL-PLUS are minor, we strongly believe that empirical coverage never lives in a vacuum. And therefore we outline the main contributions of this paper above DUAL and the other systems in the last subsection.

4.4 Key Contributions

This paper introduced a new approach to improper scope phenomena, one that can be viewed as a synthesis of E-type and dynamic approaches, combining the strengths of each. The centerpiece of the approach is a new logic, FOL-PLUS, which is a simple extension of static, first order logic. While many of the ideas presented here have analogs in prior work, we conclude the paper with what we view as the contributions of FOL-PLUS.

Formal Simplicity FOL-PLUS is a much simpler system overall than any other with comparable empirical coverage, as argued extensively above. But this simplicity can be cashed out in several ways:

- <u>FOL-PLUS is static.</u>
  Given an assignment g, a FOL-PLUS formula denotes a simple truth value, just as in FOL itself. We have shown that the key contributions of dynamic logic, even plural dynamic logic, can be replicated in a static system, which itself we take as a significant contribution. Furthermore, FOL-PLUS is not a complex "statification" converting a dynamic logic into a static one; instead, we present a conceptually simple static logic, thereby providing insight into the properties that underlie dynamic systems.

  The simplicity of a static logic over a dynamic one may seem apparent, but it also reverberates throughout the system. For instance, conjunction in FOL-PLUS is the familiar two-place truth function of FOL, while dynamic logics require a non-commutative operation equivalent to relation composition. And every separate natural language translation into dynamic logic must also take dynamic effects into account.

- FOL-PLUS operates over a single assignment.
  In general, the semantic value $[\![\phi]\!]$ that a dynamic logic assigns to a formula is a relation between *states,* where the complexity of the state depends on the choice of logic. In a singular logic like Dynamic Predicate Logic (Groenendijk and Stokhof 1991), states are single assignments, hence semantic values are relations between assignments. In plural logics following van den Berg (1996), states are sets of assignments, typically conceptualized as *tables* in which assignments correspond to rows and variables correspond to columns. Thus a semantic value (for a formula) is a relation between tables. In a static logic such as FOL, a semantic value is a function from an assignment to a truth value, which is to say, a 1-place relation over assignments (that is, a set of assignments). Thus we have at least three orders of magnitude of **state complexity:** 1-place relations over assignments (FOL), 2-place relations over assignments (DPL, DUAL[17]), and 2-place relations over tables (plural dynamic logics). FOL-PLUS belongs to the simplest class, with FOL.

- FOL-PLUS involves no non-determinism.
  The relations defined by dynamic logics are conceived as relations between inputs and outputs of a program. A program that defines a general (non-function) relation is *non-deterministic:* a given input does not determine a unique output. Such programs are intrinsically more difficult to reason about than deterministic programs—indeed, to the best of our knowledge, every widely-used practical programming language is deterministic. Nondeterministic programs can be converted to equivalent deterministic programs by increasing the state complexity: the *set* of outputs produced by a given input is uniquely defined. (This technique is fundamental in the treatment of finite-state automata, for example.) Thus what we may call the **deterministic state complexity** of standard dynamic logics is actually greater than the nondeterministic state complexity discussed in the previous paragraph. For DPL, the equivalent deterministic program defines a 2-place relation between tables, and for dynamic plural logics, it defines a 2-place relation between *sets of tables.* Non-determinism is not an issue for FOL or FOL-PLUS; their state complexity remains at 1-place relations over assignments. Thus the true increase in state complexity from static to dynamic logics is even greater than stated earlier. FOL-PLUS is the only plural logic we know of whose deterministic state complexity is that of a 1-place relation over assignments.

  Beyond the inherent unwieldiness of complex states, this too is a feature that reverberates throughout a semantic system. For instance, existential closure in a van den Berg system is far from straightforward, requiring pointwise comparison of the individual component assignments in the input and output information states.

[17]DUAL also adds an additional assignment parameter for tracking uppercase variables. Its state complexity is therefore technically a 2-place relation over pairs of assignments.

- <u>FOL-PLUS does not rely on special operators.</u>
  Our full array of empirical results is obtainable using only the formal system described in Figure 2 above and standard predicate definitions. As a comparison, in addition to analogs of FOL structures such as atomic formulas, conjunction, and so forth, Brasoveanu (2007) introduces a significant number of additional operators whose meanings are not describable in terms of standard predicates: three maximization operators, four distribution operators, two special determiner denotations for generalized quantifiers, a $\ulcorner$**unique**$\urcorner$ operator for singular items, and more.

Discourse Blocks FOL-PLUS introduces discourse blocks as the locus of three seemingly unrelated operations: the scope of indefinites, summation and existential closure, and formula-label anaphora. DUAL has the beginnings of a theory of discourse blocks, namely the labeled subformulas in a DUAL formula. However, DUAL does not capture the connection between such labeled subformulas on the one hand and variable scope and closure operators on the other.

Translation Simplicity FOL-PLUS is designed to map neatly onto natural language, allowing (for instance) quantifiers to denote simple two-place relations over (plural) individuals. Although Keshet (2018) does not give a full translation procedure for DUAL, he does point out that such a translation would perforce require rather complex, higher order denotations for quantificational determiners, juggling two uppercase indices and one lowercase index (see Keshet 2018, p. 282). A translation along these lines is illustrated in (129):

(129) $[\text{every}^{x}_{R,S}\ \beta\ \gamma] \rightsquigarrow (R : \exists x \beta') \wedge (S : R \wedge \gamma') \wedge \textsc{every}(R.x, S.x)$

And while some other plural logics provide translations from natural language (e.g. Brasoveanu 2007), many do not (e.g. Keshet 2018). The mapping we provide is a very simple extension of the familiar system due to Heim and Kratzer (1998).

A Lambda Notation

Adding the lambda operator in a fully general way entails significant complexities, but we can sidestep those complexities by making two assumptions:

(130) a. The output of translation is converted to standard form (see 1.4.1), before applying β-reduction.
b. Variables bound by λ are always fresh variables, occurring nowhere else in the discourse.

Note that lambda is not a closure operator and does not close blocks. By *lambda application* we mean an expression of form $(\lambda\alpha\phi)(\psi)$ where ϕ is the *body* and ψ

is the *argument.* Any free variables in the body or the argument belong to the block that contains the lambda application.

(130a) guarantees that all bracketed variables are immediately bound, and, in particular, that there are no free bracketed variables in a lambda application. As for bracketed variables that are free in the value of a formula label, recall that:

(131) Formula retrieval and storage are only permitted in expressions of form $C_X^Y \phi$, where C is a closure operator.

This guarantees that any bracketed variables in X are immediately bound by C.

(130b) is partly a syntactic constraint (rule PA takes the variable for λ from a syntactic index) and partly a semantic constraint (rule PM).[18] Requiring all variables to be fresh obviates the need for α-conversion. Incidentally, in our examples, we use z with optional numeric subscripts to indicate that a variable is intended to be "fresh."

Given (130), we may simplify the output of translation by (1) conversion to standard form and (2) β-reduction (without α-conversion). The resulting expression will be in standard form and will contain no lambdas.

References

C. Barker and C. Shan. Donkey anaphora is in-scope binding. *Semantics and Pragmatics*, 1(1):1–46, 2008.

J. Barwise and R. Cooper. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4(2):159–219, 1981.

Jon Barwise. Noun phrases, generalized quantifiers and anaphora. In *Generalized quantifiers*, pages 1–29. Springer, 1987.

George S. Boolos and Richard C. Jeffrey. *Computability and Logic, 2nd edition.* Cambridge University Press, 1980.

Adrian Brasoveanu. *Structured nominal and modal reference.* PhD thesis, Rutgers University New Brunswick, NJ, 2007.

Adrian Brasoveanu. Donkey pluralities: plural information states versus non-atomic individuals. *Linguistics and philosophy*, 31(2):129–209, 2008.

Adrian Brasoveanu and Donka F Farkas. How indefinites choose their scope. *Linguistics and philosophy*, 34(1):1–55, 2011.

E.F. Codd. A relational model for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

[18] The syntactic constraint is satisfied as a consequence of usual assumptions on distinctness of indices, and the semantic assumption is unremarkable.

Jan van Eijck. Axiomatising dynamic logics for anaphora. *Journal of Language and Computation*, 1(1):103–126, 1999.

Paul Elbourne. *Situations and Individuals*. Cambridge: The MIT Press, 2005. ISBN 0262050803.

Paul Elbourne. Bishop sentences and donkey cataphora: A response to Barker and Shan. *Semantics and Pragmatics*, 2(1):1–7, January 2009. doi: 10.3765/sp.2.1.

Paul Elbourne. Incomplete descriptions and indistinguishable participants. *Natural Language Semantics*, 24(1):1–43, 2016.

Gareth Evans. Pronouns, quantifiers, and relative clauses (i). *Canadian Journal of Philosophy*, 7(3):467–536, 1977. ISSN 00455091.

Gareth Evans. Pronouns. *Linguistic Inquiry*, 11(2):337–362, 1980.

J.D. Fodor and I.A. Sag. Referential and quantificational indefinites. *Linguistics and Philosophy*, 5(3):355–398, 1982.

Gerald Gazdar. A cross-categorial semantics for coordination. *Linguistics and Philosophy*, 3(3):407–409, 1980.

P.T. Geach. *Reference and generality: an examination of some medieval and modern theories*. Ithaca, NY: Cornell University Press, 1962.

J. Groenendijk and M. Stokhof. Dynamic predicate logic. *Linguistics and philosophy*, 14(1):39–100, 1991.

I. Heim. *Meaning, Use and Interpretation of Language*, meaning, use and interpretation of language File change semantics and the familiarity theory of definiteness, pages 223–248. De Gruyter, Berlin, 1983. ISBN 0470758333.

I. Heim. E-type pronouns and donkey anaphora. *Linguistics and Philosophy*, 13(2):137–177, 1990.

Irene Heim. *The Semantics of Definite and Indefinite Noun Phrases*. PhD thesis, Univeristy of Massachusetts, Amherst, 1982.

Irene Heim and Angelika Kratzer. *Semantics in Generative Grammar*. Oxford: Blackwell, 1998.

Pauline Jacobson. Paycheck pronouns, bach-peters sentences, and variable-free semantics. *Natural Language Semantics*, 8(2):77–155, 2000. ISSN 0925-854x.

N. Kadmon. *On unique and non-unique reference and asymmetric quantification*. PhD thesis, University of Massachusetts, 1987.

H. Kamp. A theory of truth and semantic representation. *Formal Semantics*, pages 189–222, 1981.

H. Kamp and U. Reyle. *From discourse to logic: Introduction to modeltheoretic semantics of natural language, formal logic and discourse representation theory*, volume 42. Kluwer Academic Dordrecht,, The Netherlands, 1993. ISBN 0792310276.

Lauri Karttunen. Pronouns and variables. In *Fifth Regional Meeting of the Chicago Linguistic Society*, pages 108–115, 1969.

Edward L. Keenan and Leonard M. Faltz. Logical types for natural language. *UCLA Occasional Papers in Linguistics*, 3, 1978.

Ezra Keshet. Dynamic update anaphora logic: A simple analysis of complex anaphora. *Journal of Semantics*, 35(2):263–303, 2018.

Angelika Kratzer and Junko Shimoyama. Indeterminate pronouns: The view from Japanese. In Yukio Otsu, editor, *The Proceedings of the Third Tokyo Conference on Psycholinguistics*, pages 1–25. Tokyo: Hituzi Syobo, 2002.

D. Lewis. *Formal semantics of natural language*, chapter Adverbs of quantification, pages 3–15. Cambridge Univ. Press, 1975.

Karen S Lewis. Discourse dynamics, pragmatics, and indefinites. *Philosophical Studies*, 158(2):313–342, 2012.

Matthew Mandelkern and Daniel Rothschild. Definiteness projection. *Natural Language Semantics*, 28(2):77–109, 2020.

R. Montague. English as a formal language. *Linguaggi nella Societa e nella Tecnica*, pages 189–223, 1970.

Reinhard Muskens. Combining montague semantics and discourse representation. *Linguistics and Philosophy*, 19(2):143–186, 1996.

Rick Nouwen. Complement anaphora and interpretation. *Journal of Semantics*, 20(1):73–113, 2003a.

Rick Nouwen. E-type pronouns: Congressmen, sheep, and paychecks. *The Wiley Blackwell Companion to Semantics*, pages 1–28, 2020.

Rick Nouwen, Adrian Brasoveanu, Jan van Eijck, and Albert Visser. Dynamic Semantics. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2022 edition, 2022.

Rick Willem Frans Nouwen. *Plural pronominal anaphora in context: Dynamic aspects of quantification.* PhD thesis, 2003b.

Tanya Reinhart. Quantifier scope: How labor is divided between qr and choice functions. *Linguistics and philosophy*, pages 335–397, 1997.

M. Rooth and B. Partee. Generalized Conjunction and Type Ambiguity. *Meaning, Use, and Interpretation of Language, de Gruyter, Berlin*, pages 361–383, 1983.

Mats Rooth. A theory of focus interpretation. *Natural Language Semantics*, 1 (1):75–116, 1992.

Lenhart K Schubert and Francis Jeffry Pelletier. Generically speaking, or, using discourse representation theory to interpret generics. In *Properties, types and meaning*, pages 193–268. Springer, 1989.

R. Schwarzschild. Singleton Indefinites. *Journal of Semantics*, 19(3):289–314, 2002.

Roger Schwarzschild. GIVENness, AvoidF and other constraints on the placement of accent. *Natural Language Semantics*, 7(2):141–177, 1999. ISSN 0925-854x.

Peter Sells. *Restrictive and non-restrictive modification*, volume 28. Center for the Study of Language and Information, Stanford University, 1985.

Martin H van den Berg. *Some aspects of the internal structure of discourse. The dynamics of nominal anaphora.* PhD thesis, Amsterdam ILLC, 1996.

J.D. Ullman. *Principles of Database and Knowledge-Base Systems, Volume 1.* Computer Science Press, Rockville, Maryland, 1998.

Yoad Winter. Choice functions and the scopal semantics of indefinites. *Linguistics and philosophy*, pages 399–467, 1997.