Compositional trace conversion*

Robert Pasternak
Leibniz-Center for General Linguistics (ZAS)

Abstract In order to eliminate traces as stipulated grammatical objects, syntactic movement has been reformulated in terms of *multiple-merge*: it is the result of the same constituent being merged into the structure multiple times, using either *copies* or *multidominance structures*. Multiple-merge theories of movement pose known challenges for the semantic interpretation of quantifier raising, as there are no variable-denoting traces in lower positions. The most common means of resolving this conundrum is *trace conversion* (Fox 2002, 2003), in which either a syntactic operation makes alterations at lower merge sites in order to generate trace-like interpretations, or the semantics behaves *as if* such a syntactic operation had occurred. In this paper, I discuss problems faced by presently formulated versions of trace conversion: the syntactic variant requires undesirable countercyclic insertion, while the semantic variant violates principles of compositionality. I propose an alternative, *compositional trace conversion*, in which multiple-merge structures can be directly interpreted in a straightforwardly compositional manner. This approach is shown to generalize well, extending to modals and degree phrases as well as DPs.

Keywords: scope, copy theory, multidominance, quantifiers, modals, comparatives, trace conversion, compositionality

1 Introduction

For much of the history of generative syntax since Chomsky 1957, *displacement*—the apparent tendency for constituents to simultaneously occupy multiple syntactic locations—has been cast in terms of *movement*: a constituent seems to occupy multiple locations because it starts in one spot and moves to another, leaving a trace. Take, for example, (1):

* For helpful discussion, many thanks to Patrick Elliott, Michael Yoshitaka Erlewine, Kai von Fintel, Nicholas Fleisher, Thomas Graf, Itamar Kastner, Uli Sauerland, Giorgos Spathas, and audience members at the ZAS Semantics and Pragmatics Reading Group, the University of Göttingen's Oberseminar English Linguistics, and the LSA 2020 Annual Meeting. Special thanks to Patrick Elliott for many long and fruitful conversations on all aspects of this project. The author's research is funded by DFG Grant #387623969 (*DP-Border*, PIs: Artemis Alexiadou and Uli Sauerland).

(1) Harvey was punched.

Harvey here seems to play two syntactic roles: it is the internal argument of *punch*, and the subject of the sentence. To account for this, *Harvey* is traditionally taken to start as the complement of *punch*, subsequently moving into subject position and leaving a trace in its initial position:

(2) Harvey_1 was punched t_1

Naturally, the same approach extends to quantifier raising (QR). Consider (3) on its inverse scope interpretation, in which teachers can covary with students:

(3) A student likes every teacher.

Here, the two positions that *every teacher* seems to simultaneously occupy are its overt position as the internal argument of *like*, and a higher position at which it outscopes *a student*. This is again captured by means of movement: *every teacher* covertly moves past *a student*, leaving behind a trace that is interpreted as a bound variable argument to *like*:

(4) $[_{\text{TP}}\ [\text{every}_2\ \text{teacher}]\ \lambda_2\ [_{\text{TP}}\ [\text{a}_1\ \text{student}]\ \lambda_1\ \text{T}\ [_{\text{VP}}\ t_1\ \text{like}\ t_2]]]$

However, the existence of traces as a distinct kind of syntactic object has come under fire in the past couple of decades. Traces have several seemingly undesirable properties, including that (I) they are stipulated as part of the language faculty instead of being derived from prior principles; (II) they are non-lexical objects inserted into syntactic computations;[1] and (III) their insertion is inherently countercyclic, since they are placed in locations vacated by moving constituents. For these and other reasons, one aim of the Minimalist Program (Chomsky 1995) has been to do away with traces, and to derive those empirical facts normally attributed to them from other grammatical operations and principles whose core motivations are clearer.

Researchers have mostly coalesced around a single broad strategy for accomplishing this. We know that some sort of structure-building operation—what Chomsky (1995) calls Merge—is needed in order to generate syntactic structures to begin with, meaning that this structure-building operation can be taken for granted as part of the language faculty. So what if, when some constituent X undergoes "movement", what actually happens is that X is simply merged back into the structure again, this time at a higher point in the tree? This eliminates the need for traces and the stipulations that come with them: displacement is not movement-plus-trace-insertion, but rather a single constituent appearing in multiple syntactic locations, by means of an operation that is independently motivated on basic conceptual grounds. I will refer to analyses in this broad program as *multiple-merge theories of movement*.

1 By "lexical objects" I do not mean to evoke the distinction between lexical and functional heads, but rather to refer to anything stored in the lexicon, whether it be "lexical" or "functional".

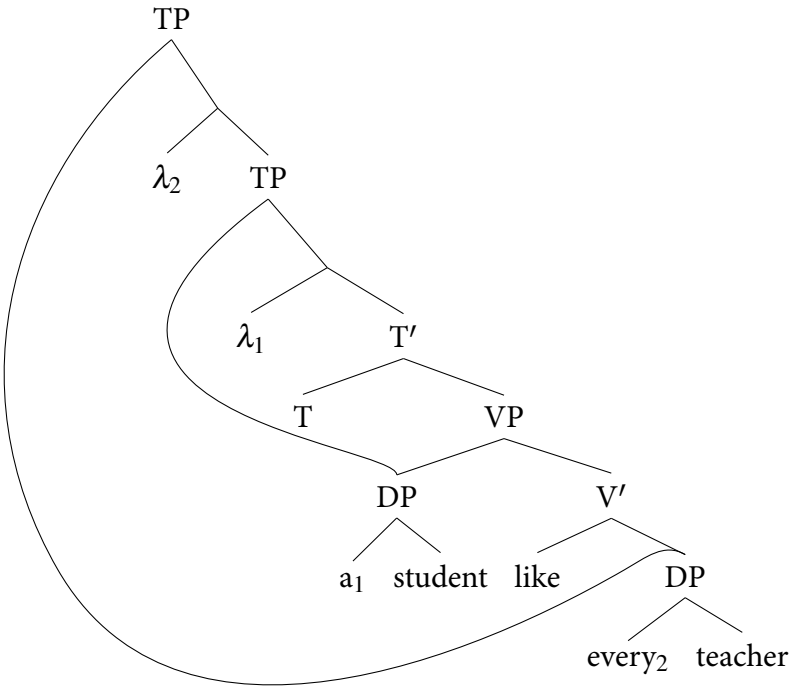

Figure 1 Multidominance LF for inverse scope of *A student likes every teacher*.

Under the multiple-merge umbrella, two main candidate theories have emerged, with the locus of disagreement being what constitutes "the same" constituent sitting in multiple locations. The first is the *copy theory of movement* (Chomsky 1995), in which a "moving" constituent is simply copied, with the newly created copy merged at the destination of movement. Thus, on the most direct translation, the trace-laden LF in (4) would be replaced with (5):

(5) $[_{TP}$ [every$_2$ teacher] λ_2 $[_{TP}$ [a$_1$ student] λ_1 T $[_{VP}$ a$_1$ student like every$_2$ teacher]]]

The second approach is the *multidominance theory of movement* (Starke 2001, Gärtner 2002, Johnson 2012). According to multidominance theories, it is possible for a single constituent to have multiple mothers, and this is precisely the configuration that arises in the case of movement: when constituent X moves from within Y to the specifier of ZP, no copies are made, and instead we end up with a structure in which X has both Y and ZP as mothers. In other words, rather than having two indistinguishable copies of X, with one sitting in Y and one sitting in spec-ZP, it is quite literally the same constituent X that sits in both positions. Thus, the translation of the LF in (4) into the multidominance theory of movement will look as in Fig. 1.

In addition to their conceptual advantages, multiple-merge theories have been shown to have empirical advantages as well. But multiple-merge theories also pose challenges for traditional approaches to semantic composition. A common means of interpreting LFs like (4) is to treat traces as denoting variables bound by the lambda-abstracting nodes λ_1 and λ_2, generating predicates that serve as arguments to their respective quantifiers (Heim & Kratzer 1998). But this view of compositionality seems untenable in the face of LFs like (5) and Fig. 1, as it would require that the DP *every teacher*, for example, be interpreted as a true quantifier at the higher merge site, and as a bound variable (more or less) at the lower merge site. Put another way, there is an apparent tension between the following principles of semantic interpretation: (I) a quantificational DP introduces quantification at its highest merge site; (II) quantificational DPs do not introduce quantification at lower merge sites, and are interpreted as bound variables; and (III) structurally identical DPs are also semantically identical, regardless of whether it's two indistinguishable copies (copy theory) or the same DP interpreted at multiple merge sites (multidominance).

The most commonly adopted means of dissolving this tension is *trace conversion*, proposed by Fox (2002, 2003) within the confines of the copy theory of movement. Trace conversion is a post-syntactic operation that replaces lower copies of determiners with bound definite determiners, which generate bound variable readings that allow for successful lambda abstraction:[2]

(6) (5) AFTER TRACE CONVERSION:

$[_{TP}$ [every$_2$ teacher] λ_2 $[_{TP}$ [a$_1$ student] λ_1 T $[_{VP}$ **the$_1$** student like **the$_2$** teacher]]]

Thus, trace conversion resolves the semantic conflict by modifying the structure of lower DP copies. As a result, the third principle above is maintained but irrelevant, since lower and higher copies will be structurally distinct, and can therefore have distinct denotations (bound variable for lower, quantificational for higher). An alternate version of trace conversion, mentioned as a possibility by Fox (2003), shifts the burden from the syntax to the semantics: there is no syntactic operation that modifies lower copies, but instead the compositional semantics interprets multiple-merge structures *as if* some such syntactic operation had taken place.

In Section 2 I will discuss some crucial downsides to both syntactic and semantic trace conversion as they are typically formulated, based on a mixture of empirical observations, theory-internal considerations, and broader principles of theoretical simplicity and elegance. In brief, syntactic trace conversion requires the counter-cyclic insertion of interpreted material (contrary to the so-called *Inclusiveness Condition*, Chomsky 1995), as well as additional stipulations to account for non-DP scope-taking, while semantic trace conversion runs afoul of basic principles of semantic compositionality. In addition to arguing against syntactic and semantic trace

2 For a different but thematically similar idea, see Sauerland 1998, 2004.

conversion as they are presently formulated, I will also briefly discuss an alternative approach, proposed by Johnson (2012) and extended by Fox & Johnson (2016), that resolves these issues by fundamentally revising the architecture of natural language quantification: determiners do not actually quantify, and the quantificational force traditionally associated with them stems instead from the semantics of a separate head merged along the clausal spine. However, I will not argue against a Johnson-style approach in detail, instead focusing on improving trace conversion in a way that does not require substantial changes to traditional assumptions about quantification. Such improvements are nonetheless relevant to debates about the relative merits of Johnson's approach and trace conversion-based approaches: after all, the goal of the debate is to determine which theory is superior *in its best form*, so isolated improvements to either theory can generate their own insights and help avoid quibbles over what turn out to be accidental traits of a given analysis.

In Section 3 I turn to my own analysis, which I refer to as *compositional trace conversion*, and which allows LFs like (5) and Fig. 1 to be interpreted compositionally (unlike semantic trace conversion), without syntactic modifications (unlike syntactic trace conversion). Put simply, the semantic impact of trace conversion—that is, the "swapping out" of a quantificational interpretation for a bound definite interpretation at lower merge sites—is automatically triggered by the operation of lambda abstraction. In Section 4, I show how this analysis can easily be type-generalized and thus extended beyond DP quantification, accounting for the scope-taking behavior of both modals and degree phrases in comparatives. I offer some concluding remarks and lines for potential future research in Section 5.

2 Trace conversion and its discontents

In this section I will discuss syntactic and semantic trace conversion in greater detail, as well as the critical issues that each version faces. This will pave the way for my own analysis later in the paper, which generates the semantic effects of trace conversion while avoiding those problems faced by prior implementations. I will also briefly discuss an alternative proposal by Johnson (2012) and Fox & Johnson (2016) that seeks to resolve these issues by abandoning the traditional assumption that determiners are responsible for the quantification typically associated with them.

2.1 Syntactic trace conversion

The most popular approach to trace conversion is *syntactic*: a post-syntactic operation replaces quantificational determiners at lower copies with the bound definite determiner *the*. Thus, (5), which is the LF for (3) generated in the copy theory of movement, is converted to (6):

(5) $[_{TP}$ [every$_2$ teacher] λ_2 $[_{TP}$ [a$_1$ student] λ_1 T $[_{VP}$ a$_1$ student like every$_2$ teacher]]]

(6) $[_{TP}$ [every$_2$ teacher] λ_2 $[_{TP}$ [a$_1$ student] λ_1 T $[_{VP}$ **the$_1$** student like **the$_2$** teacher]]]

Now suppose that $[\![\text{the}_n]\!]$, evaluated with respect to a variable assignment g, takes a predicate P and returns $g(n)$ if $g(n)$ is a P, and is otherwise undefined. In other words, it takes a predicate and returns a *restricted variable*:

(7) $[\![\text{the}_n]\!]^g = \lambda P : P(g(n)).\ g(n)$

Thus, $[\![\text{the}_1 \text{ student}]\!]^g$ denotes $g(1)$ iff $g(1)$ is a student (and is otherwise undefined), and $[\![\text{the}_2 \text{ teacher}]\!]^g$ denotes $g(2)$ iff $g(2)$ is a teacher. In this case, we get precisely the bound variable reading we desired for lower copies, since lambda abstraction can subsequently bind these restricted variables without a hitch. Meanwhile, the higher copies, which were not altered by trace conversion, receive the same quantificational interpretation they are typically assigned. In other words, we get essentially the same interpretation as on more traditional, trace-based approaches (since traces are generally taken to denote variables), but without having to posit traces as a syntactic object furnished by the innate language faculty. Moreover, what trace conversion swaps in for quantificational determiners at lower copies—the bound definite determiner—is a type of syntactic object whose existence has already been motivated on entirely independent grounds: simply put, there is such a thing as overt *the*. Thus, not only does syntactic trace conversion follow through on the Minimalist Program's avoidance of traces while allowing for straightforward semantic computation, but it does so while also only making use of syntactic objects whose inclusion as a part of the grammar has been independently motivated.

However, syntactic trace conversion is not without its drawbacks. Notice that trace conversion is an operation performed only on lower copies of a DP, since the highest copy must retain its quantificational interpretation. But without any recourse to look-ahead there is no way of telling that a given copy is a lower copy until movement has already taken place, at which point the lower copy is already embedded in a larger structure.[3] In other words, trace conversion is inherently countercyclic. If it were simply a deletion operation, this would not necessarily be problematic, since countercyclic deletion operations at the interfaces are far from unprecedented (e.g., PF deletion theories of ellipsis). But syntactic trace conversion not only deletes lower-copy quantifiers, but also inserts in their place—again, countercyclically—semantically interpreted lexical material absent from the numeration (*the*), in violation of Chomsky's (1995) otherwise robust Inclusiveness Condition (IC). While IC is an empirical hypothesis, and thus should not be taken as gospel, a grammar

3 *Look-ahead* refers to a hypothetical feature of grammars in which the (non)occurrence of some operation can in some sense depend on what happens later in the derivation. Natural language syntax is generally thought to lack look-ahead.

that violates it is less optimal than a grammar that does not (all else being equal), so on basic minimalist principles the prospect of abandoning IC should give us pause. This is especially true if the only motivation for abandoning IC is the preservation of what might in other circumstances be seen as a quite specific semantic analysis of quantification. Put differently, if the received view of quantifier semantics happened to be one that meshed well with multiple-merge theories of movement, there would be no reason to posit syntactic trace conversion in the first place.

This version of syntactic trace conversion also suffers from the fact that DPs are not the only type of syntactic constituent that takes scope by means of movement. For instance, degree phrases in comparatives can give rise to scope ambiguities, as evidenced by examples (8) and (9) from Heim (2000: 48, paraphrases mine), in which the degree phrases *exactly 5 pages -er than that* and *less than that* can scope either above or below the intensional verb *require*:

(8) (This draft is 10 pages.) The paper is required to be exactly 5 pages longer than that.
   a. The paper must be exactly 15 pages. (*require* > DegP)
   b. The minimum length is precisely 15 pages. (DegP > *require*)

(9) (This draft is 10 pages.) The paper is required to be less long than that.
   a. The maximum length is under 10 pages. (*require* > DegP)
   b. The minimum length is under 10 pages. (DegP > *require*)

In addition, Iatridou & Zeijlstra (2013) argue that the relative scope of modals and negation is resolved by means of movement. More specifically, they argue that modals are merged under negation and move overtly past it; modals like *can* that scope under negation then reconstruct to their pre-movement positions, while those like *must* that scope over negation are interpreted in their post-movement positions, leaving a trace—or, assuming the copy theory of movement, a copy—in their merge positions.

(10) a. Rivka cannot leave the party. (LF: [not [can…]])
   b. Rivka must not leave the party. (LF: [must$_1$ λ_1 [not [t_1…]]])

Thus, according to a straightforward implementation of trace conversion, degree morphemes like *-er* and *less* and modals like *must* have to be covertly replaced with bound definite determiners at lower copies.[4] While one can of course simply bite the bullet and accept that modals and degree heads (and other scope-taking heads) can be replaced at LF with definite determiners, an operation that covertly inserts *the* in syntactic environments in which it cannot overtly appear is at face value an undesirable one to posit, at least without substantial further motivation.

4 Heim (2006) argues that *less* is not monomorphemic, but is composed of *-er* + *little*, with *-er* doing the degree quantification. In this case, for both (8) and (9) it is *-er* that must be replaced with *the*.

A reasonable response to this empirical problem would be to posit that syntactic trace conversion does not insert the actual lexical determiner *the*, but rather something else that has a similar semantic contribution. Moulton (2015), for example, adopts such a view, proposing a rule of *Category-Neutral Trace Conversion (CNTC)*:

(11) Category-Neutral Trace Conversion (Moulton 2015: 326):
   a. Quantifier Removal: $[_{\text{DP}}$ every square$]_3 \rightsquigarrow [_{\text{DP}}$ square$]_3$
   b. Index Interpretation: $[_{\text{DP}}$ square$]_3 \rightsquigarrow [_{\text{DP}}$ 3: 3 is a square$]$

He then posits that "the output of Index Interpretation is shorthand for the semantics, which interprets the index as a restricted variable" (Moulton 2015: 326):

(12) Semantic Interpretation Post-CNTC (Moulton 2015: 326):
   $[\![[_{\text{DP}}$ 3: 3 is a square$]]\!]^g = g(3)$ iff $[\![\text{square}]\!](g(3)) = 1$; undefined otherwise

But even though CNTC avoids the problem of determiners in places they do not belong, and even though Quantifier Removal itself is innocent enough (being a deletion operation), it remains the case that Index Interpretation is a syntactic operation that countercyclically inserts non-enumerated, semantically interpreted material. In addition, while Moulton does not go through how the output of Index Interpretation is interpreted compositionally, it seems as though an altogether novel rule of semantic interpretation is required, along the following lines:

(13) $[\![[_{\text{XP}}\ n : n \text{ is Y}]]\!]^g = g(n)$ iff $[\![\text{Y}]\!](g(n)) = 1$; undefined otherwise

An alternate version of CNTC might avoid the stipulation of a new rule of semantic composition as follows: Index Interpretation inserts some object that is distinct from *the*—call it *schme*—that is syntactically category-neutral and semantically behaves like a type-generalized bound definite:

(14) $[\![\text{schme}_n]\!]^g = \lambda J : J(g(n)).\ g(n)$

This would give us a version of trace conversion that is category-neutral, but that does not require the stipulation of a novel rule of semantic composition. But in addition to our continued violation of the Inclusiveness Condition, there is another reason why *schme* should give us pause. Recall that one advantage of the original formulation of trace conversion—that is, the one that used *the* instead of *schme*—is that we already know that *the* exists, so we are only using syntactic objects whose existence can be independently justified. But there is no such justification for *schme*, since no language has an overt definite "determiner" that cuts across semantic types and syntactic categories. So what we are left with is a newly stipulated syntactic object whose existence cannot be justified on independent grounds, and that can only be inserted countercyclically at locations from which movement originates. In other words, once syntactic trace conversion is appropriately extended to account for QR

of non-DPs, what we end up with is something that bears a suspicious resemblance to traces, which are the very thing that multiple-merge theories of movement have been trying to eliminate in the first place.

To summarize, an important advantage of syntactic trace conversion is that it can be formulated in a way that requires no alterations to the semantic apparatus, and that obeys the three intuitive semantic principles discussed in the introduction: quantificational DPs quantify, lower copies do not quantify, and identical DPs have identical semantic interpretations. However, this comes at the cost of some crucial syntactic stipulations, and in particular the countercyclic insertion of semantically interpreted material, contrary to the Inclusiveness Condition. Moreover, once trace conversion is extended beyond DP quantification, we have to either permit *the* to be inserted in syntactic environments in which it otherwise cannot appear, or we have to replace it with a newly stipulated syntactic object (which I have called *schme*) that has no overt counterpart, and that closely resembles the very thing multiple-merge theories of movement seek to replace: namely, traces.

2.2 Semantic trace conversion

An alternate version of trace conversion mentioned by Fox (2003) states that it is not syntactic but semantic: no alterations occur at LF, but the semantics interprets quantificational DPs at lower merge sites *as if* some syntactic alteration had taken place, and thus differently from at the highest merge site:

(15) Semantic Trace Conversion (Fox 2003: 110):

In a structure formed by DP movement, $\text{DP}_n[_\phi \ldots \text{DP}_n \ldots]$, the derived sister of DP, ϕ, is interpreted as a function that maps an individual, x, to the meaning of $\phi[x/n]$.

$\phi[x/n]$ is the result of substituting every constituent with the index n in ϕ with him_x, a pronoun that denotes the individual x.

Thus, unlike syntactic trace conversion, semantic trace conversion resolves the tension between the above three semantic principles by essentially abandoning the third: structurally identical DPs do not receive identical interpretations, since at lower merge sites they are interpreted as bound variables, and at higher merge sites they are interpreted as quantifiers.

Given that on a semantic trace conversion account an LF like (5) or Fig. 1 can be directly interpreted without intervening syntactic operations, this approach naturally avoids the pitfalls of its syntactic counterpart, e.g., there is no countercyclic insertion. However, in the end these problems are not so much eliminated as they are shifted from the syntax to the semantics, as we face a new problem of anti-compositionality: the semantic interpretation of a scope-bearing constituent is no

longer a function of its structure, since the same phrase is interpreted like a pronoun in some syntactic environments and like a true quantifier in others.

As a reviewer points out, this is not necessarily anti-compositional if we define compositionality *derivationally*, so that semantic computation makes use of not only the LF representation, but also information about how that LF structure was derived. Semantic trace conversion would not be in violation of this looser notion of compositionality, since lower copies would be derivationally distinct from higher copies in spite of being representationally identical, meaning that the two copies could receive distinct interpretations. But without further constraints, this grants more power to the semantic apparatus than has otherwise been empirically motivated. For example, there is no natural language determiner *nonce* that has a different interpretation depending on whether or not it undergoes, say, raising:

(16) $[\![\text{nonce}]\!] = \begin{cases} \forall & \text{if } \textit{nonce} \text{ undergoes raising} \\ \exists & \text{otherwise} \end{cases}$

(17) a. Nonce dog seems to be here. (≈ **Every** dog seems to be here.)
b. Nonce dog is here. (≈ **A** dog is here.)

Similarly, there is no pronoun *faux* whose referent must be animate if it undergoes A′-movement, or inanimate if it does not:

(18) $[\![\text{faux}_n]\!]^g$ is defined iff either (I) *faux* undergoes A′-movement and $g(n)$ is animate, or (II) *faux* does not undergo A′-movement and $g(n)$ is inanimate. Where defined, $[\![\text{faux}_n]\!]^g = g(n)$.

(19) a. Faux, I like. (≈ **Her**, I like.)
b. I like faux. (≈ I like **it**.)

Though these and similar observations are often left tacit, it is difficult to overstate their importance to semantic inquiry. Traditional Montagovian representation-only approaches to compositionality immediately predict these observations, but approaches that incorporate a mixture of derivational and representational information do not: if identical constituents with different derivational histories count as distinct as far as compositionality is concerned, without freshly stipulated restrictions all bets are off. Thus, the available evidence points to the exclusion of derivational histories from the lexical and compositional semantics, and the compositionality problem faced by semantic trace conversion must be taken seriously.

In summary, semantic trace conversion avoids the syntactic stipulations of syntactic trace conversion by shifting the work from the syntax to the semantics. However, this comes at the cost of either a partial abandonment of the principle of compositionality, or a weakening of the notion of compositionality in a way that has not been otherwise motivated. It is worth noting that the theory endorsed in this

paper bears a close resemblance to semantic trace conversion, in that the compositional apparatus is responsible for generating the semantic result of trace conversion. However, it differs by virtue of achieving this in a directly compositional manner. It thus might be thought of as a fully compositional implementation of semantic trace conversion. Hence, *compositional trace conversion*.

2.3 A brief note on an alternative

Before introducing compositional trace conversion, it is worth briefly discussing an alternative proposed by Johnson (2012) and expanded upon by Fox & Johnson (2016). Johnson argues on independent grounds in favor of multidominance instead of copies, but also notes that multidominance structures are incompatible with syntactic trace conversion: there is no distinction between "higher" and "lower" copies, so one cannot perform a syntactic operation only on lower copies. Presumably wishing to avoid the compositionality issues faced by semantic trace conversion, Johnson seeks an alternative that is strictly cyclic and semantically compositional. He proposes to achieve this goal and resolve the tension between the three guiding semantic principles above through substantial modifications to the traditionally assumed architecture of natural language quantification.

We start from the following premise: suppose, contrary to traditional assumptions, that there is no such thing as a quantificational determiner head *every*. What ends up being pronounced as *every teacher* actually starts its life as *the teacher*, with *the* contributing the same restricted variable interpretation as in syntactic approaches to trace conversion. Meanwhile, the quantification traditionally associated with *every* actually stems from a separate quantificational head $\forall$, merged wherever the universal quantification is to take scope. QR, rather than involving the whole DP undergoing covert movement, instead involves only the restrictor moving (or "moving") from the complement of *the* to the restrictor of $\forall$, thereby restricting both. A somewhat simplified version of this analysis is illustrated in Fig. 2.

If $[\![\forall]\!]$ is the denotation traditionally assigned to *every*, the LF in Fig. 2 generates the desired truth conditions. The fact that this sentence is pronounced with *every* instead of *the* is then determined by some morphological or phonological operation: for Johnson (2012), *the* and $\forall$ are linearized adjacent to each other and undergo morphological fusion, while for Fox & Johnson (2016) "[t]he definite determiner is pronounced as *every*, rather than *the*, because it is the exponent of a universal quantifier that is introduced higher" (Fox & Johnson 2016: p. 2).

This analysis cleverly avoids the problems of Inclusiveness violation and anti-compositionality faced by trace conversion. On the syntactic side, the derivation proceeds without any countercyclic operations inserting material into already-embedded

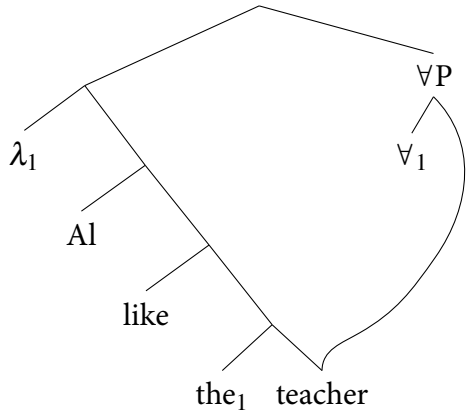

Figure 2 Johnson (2012)-style LF for *Al likes every teacher*. (Movement treated as rightward for ease of reading.)

structures, unlike syntactic trace conversion. On the semantic side, composition proceeds straightforwardly, obeying strict semantic compositionality.

My goal in this paper is not to argue against Johnson's analysis and in favor of (compositional) trace conversion. Of course, comparing and contrasting these approaches is of great concern to empirical inquiry on quantifier scope, and attempts to engage in this debate need not be hampered by the fact that the two theories are still in their developmental stages. However, it is worth emphasizing that the endgame of such comparisons is to determine which alternative is superior *in its best form*: is the best version of Johnson's approach better than the best version of trace conversion, or vice versa? Thus, value can be added to the debate simply by bringing one theory closer to its "best self", and that is what my analysis seeks to do for trace conversion. In other words, for proponents of Johnson-style approaches, the present paper can be thought of as an attempt to bridge the gap between the two theories from the side of trace conversion: while Johnson's analysis was fully syntactically and semantically compositional from the get-go, the analysis in this paper imports these features to trace conversion as well.

Naturally, gaps between the two theories must be bridged from both sides: various empirical phenomena have been addressed within traditional theories (including trace conversion) that have yet to be addressed using Johnson-style approaches. In some cases, how the latter is to be accomplished is not entirely obvious. As just one example, consider Scope Economy. Fox (2000) famously observes that while (3) is ambiguous between a surface and inverse scope reading, that ambiguity seemingly evaporates with certain instances of ellipsis such as (20), which permits only surface scope:

(20) A student likes every teacher. Mary does, too.

That this is not due simply to the presence of ellipsis can be seen by the fact that (21) retains the scope ambiguity:

(21) A student likes every teacher. An administrator does, too.

Fox derives these observations from two principles. The first is Parallelism, which states that such ellipsis can only occur if the sentences with the antecedent and elided VPs (or *v*Ps) have isomorphic LF structures. In other words, if one sentence has an inverse scope LF, the other must as well. The second is Scope Economy, which forbids semantically vacuous covert scopal operations.[5] Now suppose that for a given sentence, the "default" LF of that sentence—that is, the one that arises with only obligatory scopal operations—is one that derives surface scope, meaning that extra covert operations must apply in order to derive inverse scope. In this case, Scope Economy dictates that any derivation of the second clause in (20) in which *every* scopes over *Mary* is illicit: it would require a semantically vacuous covert operation, since *every* > *Mary* is truth-conditionally equivalent to *Mary* > *every*. Thus, the second clause in (20) must have a surface scope LF. Thanks to Parallelism, this means that the first clause must also have a surface scope LF, even though it is in principle scopally ambiguous, so no ambiguity arises in (20). This does not apply in (21), since both clauses have indefinite DPs as subjects, meaning that any covert operation that generates inverse scope will be semantically non-vacuous: the two clauses can have well-formed isomorphic inverse scope LF representations.

On Johnson's analysis, whether inverse or surface scope is derived depends not on the presence or absence of some covert scope-shifting operation, but rather on whether ∀ is merged above or below the scope site of the subject. Therefore, if we wish to introduce Scope Economy to a Johnson-style analysis, we must block derivations in which a direct object's ∀ is merged in a position higher than a referential subject (thereby making surface scope obligatory in such cases). But notice that no semantically vacuous operations are needed in order to derive illicit vacuous inverse scope: the merging of ∀ above the subject is semantically non-vacuous (it introduces universal quantification), as is the movement of the restrictor to the complement of ∀ (required for a well-formed interpretation, since otherwise ∀ will not have a restrictor). Thus, the ill-formedness of an inverse scope LF with a proper name in subject position—needed in order to capture the effects in (20) and (21)—cannot derive from the presence of a semantically vacuous covert operation, and must come from something else. One possibility is to adopt a principle like (22):

5 Mayr & Spector (2010) propose a generalized version of Scope Economy, in which covert scopal operations not only cannot be vacuous, but also cannot strengthen the interpretation. In other words, the resulting interpretation after a covert scopal operation must be either weaker or truth-conditionally independent. The point I am making applies to either version.

(22) An LF representation Z is ill-formed if there is an LF representation Z′ identical to Z except that ∀ is merged lower, and Z and Z′ have identical truth conditions.

This then predicts that what we have been calling Scope Economy is not an economy principle *per se*, since the derivations being compared are identical in terms of complexity, and differ only in their order of operations. This in turn raises questions about what the inherent motivations and empirical predictions would be for a grammatical principle like (22), and how they contrast with traditional formulations of Scope Economy. This is clearly an important area for comparison and contrast between the two types of analysis.

In short, then, while traditional approaches (including trace conversion) and Johnson-style approaches are direct competitors, and thus direct comparisons and contrasts between the two should be noted wherever they can be found, there is also significant value to be had in refining each approach individually. Ideally, this will help to highlight further areas of similarity and difference between them, as well as to avoid centering the empirical debate on what are actually contingent features of the two analyses.

2.4 Summary

To summarize, here is where we stand. Syntactic trace conversion is problematic on the grounds that it requires countercyclic insertion of semantically interpreted material, contrary to the Inclusiveness Condition. It also seems to require certain undesirable stipulations once we look beyond DP quantification: either *the* can be inserted in places where it cannot appear overtly, or we have to posit an inserted head *schme* that has many of the undesirable traits that multiple-merge theories of movement seek to avoid in the first place. Semantic trace conversion, meanwhile, faces problems of compositionality: the interpretation of a scope-taking constituent is no longer a function of its parts, since the same structure is interpreted like a bound variable at one merge location and like a true quantifier at another.

One means to a resolution, adopted by Johnson (2012) and Fox & Johnson (2016), is a wholesale revision of the presumed architecture of natural language quantification: "quantificational" heads do not quantify, and the quantification associated with them actually stems from a separate head. In the rest of this paper I will propose an alternative. In line with traditional assumptions and in contrast to (Fox &) Johnson's analysis, on my proposal determiners (and other quantificational heads) are themselves responsible for the quantification associated with them, without the need for a separate quantificational head. Unlike syntactic trace conversion, no post-syntactic operation of countercyclic insertion is required in order for the compositional semantics to work out, and the LF structures in (5) and Fig. 1 can be interpreted

directly. And unlike semantic trace conversion, the semantic composition can be derived through straightforward compositional principles, with a given constituent having the same interpretation at each merge site. Moreover, since the semantics of trace conversion is achieved strictly by compositional semantic principles, once sufficiently type-generalized the proposal will extend equally well to non-DP movement, since there is no need to insert *the* (or *schme*) in places where it does not belong.

3 Swap states and DP scope-taking

We first lay out the proposal for DP quantification. Notice that as far as the compositional semantics is concerned, the copy LF in (5) and the multidominance LF in Fig. 1 are identical: there is no compositional difference between (I) distinct but indistinguishable copies at separate merge sites, and (II) the same constituent merged at two separate locations. Thus, any analysis that can interpret one LF directly can interpret either LF directly, and this applies to the analysis proposed in this paper. For the sake of concreteness and simplicity, I will generally adopt the language of the copy theory of the movement, and will explicitly discuss things in terms of multidominance only when there are important differences between the two theories.

In going over how the system works for DP quantifier interpretation, we will use the LF in (5), repeated below, as our example:

(5) $[_{\text{TP}}\ [\text{every}_2\ \text{teacher}]\ \lambda_2\ [_{\text{TP}}\ [\text{a}_1\ \text{student}]\ \lambda_1\ \text{T}\ [_{\text{VP}}\ \text{a}_1\ \text{student like every}_2\ \text{teacher}]]]$

In order for our compositional semantics to work, we need some mechanism that will allow us to "swap out" a determiner's quantificational interpretation for a bound variable-like interpretation at lower copies, but in a straightforwardly bottom-up compositional fashion. In order to do this I will make use of what I will call *swap states*, or *states* for short. A swap state is a function that first takes an index n, then what I will call an *etett*—any function of type $(et)(et)t$, the traditional type of quantificational determiners—and returns a (possibly identical) etett.[6] That makes a swap state the somewhat cumbersome type $n((et)(et)t)((et)(et)t)$, which I will abbreviate as s. I will also use s as a variable over swap states. For a given state s, index n, and etetts D and D', if $s(n)(D) = D'$, I will say that *s swaps out D for D' at (index) n*, or equivalently, *s swaps in D' for D at (index) n*. For readability's sake, I will rewrite $s(n)(D)$ as $[D]^s_n$.

So now that we have swap states, how are they actually used? In short, they serve a role analogous to variable assignments in approaches like that of Heim & Kratzer (1998). Tradition has it that variable assignments are a parameter of semantic inter-

6 A note on notation: type $\alpha\beta$ is what is traditionally written as $\langle\alpha,\beta\rangle$. Types are right-associative, so $\alpha\beta\gamma$ is what would traditionally be written as $\langle\alpha,\langle\beta,\gamma\rangle\rangle$, while $(\alpha\beta)\gamma$ is the same as $\langle\langle\alpha,\beta\rangle,\gamma\rangle$.

pretation, and lambda abstraction returns a predicate true of an individual iff the pre-abstraction interpretation is true relative to a suitably altered variable assignment. A version of this is presented in (23):

(23) Traditional Lambda Abstraction: (cf. Heim & Kratzer 1998)
$[\![\lambda_n \text{ X}]\!]^g = \lambda x.\ [\![\text{X}]\!]^{g[n,x]}$,
where $g[n,x]$ is the g' identical to g except that $g'(n) = x$.

Similarly, in the approach presented in this paper, interpretations are parameterized to swap states—though see later discussion for some caveats—and lambda abstraction generates a predicate true of an individual iff the pre-abstraction interpretation is true relative to a suitably altered swap state. A preview of what this will look like, with important gaps to be filled in later, can be seen in (24):

(24) New Lambda Abstraction (Preview):
$[\![\lambda_n \text{ X}]\!]^s = \lambda x.\ [\![\text{X}]\!]^{s[n,?]}$,
where $s[n,?]$ is the s' such that…

The plan is that whatever $[\![\text{X}]\!]^{s[n,?]}$ looks like, it will perform the semantic work typically assigned to the operation of trace conversion.

To see how all of this works, let us start by building up the pre-abstraction VP. As always, we begin our bottom-up derivation by defining our lexical items. The denotations of *teacher* and *student* are as one might expect: they are state-insensitive *et*-type predicates. These can be seen in (25):

(25) a. $[\![\text{teacher}]\!]^s = \lambda x.\ \text{teacher}(x)$
b. $[\![\text{student}]\!]^s = \lambda x.\ \text{student}(x)$

I will often rewrite "$\lambda x.\ \text{teacher}(x)$" as the metalanguage object "teacher" when convenient, and likewise for "$\lambda x.\ \text{student}(x)$". As for $[\![\text{like}]\!]^s$, this is again more or less as one would expect, except that it must be assigned a higher type in order to allow it to directly compose with two $(et)t$-type quantificational arguments. This leads to the definition in (26):

(26) $[\![\text{like}]\!]^s = \lambda Q_{(et)t} \lambda Q'_{(et)t}.\ Q'(\lambda x.\ Q(\lambda y.\ \text{like}(x,y)))$

This just leaves us with the determiners a_1 and $every_2$, and these are where swap states make their appearance in the lexical semantics. Let **some** be the traditional existential etett (i.e., $\lambda P \lambda P'.\ P \cap P' \neq \varnothing$), and likewise for **every** and the universal etett ($\lambda P \lambda P'.\ P \subseteq P'$). Instead of simply being **some**, $[\![\text{a}_1]\!]^s$ will be whatever etett s swaps in for **some** at index 1; similarly, $[\![\text{every}_2]\!]^s$ will be whatever etett s swaps in for **every** at index 2:

(27) a. $[\![\text{a}_n]\!]^s = [\textbf{some}]^s_n = \lambda P_{et} \lambda P'_{et}.\ [\textbf{some}]^s_n(P)(P')$

b. $[\![\text{every}_n]\!]^s = [\textsc{every}]^s_n = \lambda P_{et} \lambda P'_{et}.\ [\textsc{every}]^s_n(P)(P')$

Composing our VP involves straightforward function application. First we combine $[\![\text{every}_2]\!]^s$ with $[\![\text{teacher}]\!]^s$, and then feed the result to $[\![\text{like}]\!]^s$:

(28) a. $[\![\text{every}_2]\!]^s([\![\text{teacher}]\!]^s) = \lambda P'.\ [\textsc{every}]^s_2(\text{teacher})(P')$

b. $[\![\text{like}]\!]^s([\![\text{every}_2\ \text{teacher}]\!]^s)$
$= \lambda Q'.\ Q'(\lambda x.\ [\![\text{every}_2\ \text{teacher}]\!]^s(\lambda y.\ \text{like}(x,y)))$
$= \lambda Q'.\ Q'(\lambda x.\ [\textsc{every}]^s_2(\text{teacher})(\lambda y.\ \text{like}(x,y)))$

Next we combine $[\![\text{a}_1]\!]^s$ with $[\![\text{student}]\!]^s$ and feed the result to $[\![\text{like every}_2\ \text{teacher}]\!]^s$:

(29) a. $[\![\text{a}_1]\!]^s([\![\text{student}]\!]^s) = \lambda P'.\ [\textsc{some}]^s_1(\text{student})(P')$

b. $[\![\text{like every}_2\ \text{teacher}]\!]^s([\![\text{a}_1\ \text{student}]\!]^s) = 1$ iff
$[\textsc{some}]^s_1(\text{student})(\lambda x.\ [\textsc{every}]^s_2(\text{teacher})(\lambda y.\ \text{like}(x,y)))$

And just like that, our VP is composed.

Treating tense (T) as semantically vacuous for simplicity's sake, the next step is lambda abstraction via λ_1. As mentioned above, this entails manipulating the swap state in order to swap out the quantificational etett **SOME** for a bound variable etett. In order to do this, though, we must know what our bound variable etett is. Whatever it is, it must be parameterized to an individual: namely, the entity argument z that is lambda-abstracted over. Three immediate options for this bound variable etett spring to mind, defined below:

(30) a. $\textsc{bd}_z := \lambda P \lambda P'.\ P(z) \wedge P'(z)$

b. $\textsc{bd}'_z := \lambda P \lambda P'.\ P'(z)$

c. $\textsc{the}_z := \lambda P \lambda P' : P(z).\ P'(z)$

If $\textsc{bd}_z$ is swapped in for a quantificational etett, the prediction is that both the restrictor and the scope of a lower copy are interpreted, and both contribute assertive content. Meanwhile, if $\textsc{bd}'_z$ is swapped in for a quantificational etett, we predict that the restrictors of lower copies make no semantic contribution whatsoever: λP binds nothing. Finally, $\textsc{the}_z$, which would generate results identical to those of trace conversion as implemented by Fox (2002, 2003), is like $\textsc{bd}_z$ in that it treats the restrictors of lower copies as making semantic contributions, but is unlike $\textsc{bd}_z$ in that the restrictor is presuppositional rather than assertive.[7]

So which of these three is right? Sauerland (1998, 2004) argues convincingly and at length in favor of the hypothesis that the restrictors of lower copies of DPs do

7 For those familiar with Fox's work, the equivalence between my $\textsc{the}_z$ and Fox's trace conversion may not be obvious. However, note that $\textsc{the}_z$ is equivalent to $\lambda P \lambda P'.\ P'(\iota x[P(x) \wedge x = z])$, a type-lifted version of the semantic result of Fox's trace conversion.

indeed make semantic contributions, based on a variety of facts pertaining to ellipsis. If Sauerland is right, then this rules out $\textbf{BD}'_z$, but the choice between $\textbf{BD}_z$ and $\textbf{THE}_z$ is still not obvious. I will opt for $\textbf{THE}_z$ in keeping with Fox's analysis, but leave open the possibility that $\textbf{BD}_z$ is the right choice.

Now that we have decided which bound variable etett to swap in for **SOME** when lambda abstracting over index 1, we next need to decide on how to actually perform this swap. In the traditional lambda abstraction in (23), this is done by replacing the variable assignment g with a variable assignment $g[1,z]$ that is identical to g except that $g[1,z](1) = z$. Similarly, for us this will involve replacing the swap state s with the state $s[1,\textbf{THE}_z]$, which is the state identical to s except that $s[1,\textbf{THE}_z]$ swaps out all etetts for $\textbf{THE}_z$ at index 1. More generally:

(31) $$s[n,D] := \lambda n' \lambda D'.\ \begin{cases} D & \text{if } n' = n \\ s(n')(D') & \text{if } n' \neq n \end{cases}$$

With this in place, we now have the formal tools necessary in order to define lambda abstraction and fill in the blanks in (24). This can be seen in (32):

(32) New Lambda Abstraction:
$[\![\lambda_n\ \mathrm{X}]\!]^s = \lambda z.\ [\![\mathrm{X}]\!]^{s[n,\textbf{THE}_z]}$

Turning back to our example, the result of lambda abstraction is as follows:

(33) $[\![\lambda_1\ \mathrm{a}_1\ \text{student like every}_2\ \text{teacher}]\!]^s$
$= \lambda z.\ [\![\mathrm{a}_1\ \text{student like every}_2\ \text{teacher}]\!]^{s[1,\textbf{THE}_z]}$
$= \lambda z.\ [\textbf{SOME}]_1^{s[1,\textbf{THE}_z]}(\text{student})$
$(\lambda x.\ [\textbf{EVERY}]_2^{s[1,\textbf{THE}_z]}(\text{teacher})(\lambda y.\ \text{like}(x,y)))$

By definition, for any state s and etett D, $[D]_1^{s[1,\textbf{THE}_z]} = \textbf{THE}_z$, meaning that $[\textbf{SOME}]_1^{s[1,\textbf{THE}_z]}$ can be replaced with $\textbf{THE}_z$. Similarly, for any state s, etett D, and $n \neq 1$, $[D]_n^{s[1,\textbf{THE}_z]} = [D]_n^s$, meaning that $[\textbf{EVERY}]_2^{s[1,\textbf{THE}_z]}$ can be replaced with $[\textbf{EVERY}]_2^s$.

(34) $[\![\lambda_1\ \mathrm{a}_1\ \text{student like every}_2\ \text{teacher}]\!]^s$
$= \lambda z.\ \textbf{THE}_z(\text{student})(\lambda x.\ [\textbf{EVERY}]_2^s(\text{teacher})(\lambda y.\ \text{like}(x,y)))$
$= \lambda z : \text{student}(z).\ [\textbf{EVERY}]_2^s(\text{teacher})(\lambda y.\ \text{like}(z,y))$

We now have an *et*-type predicate, which naturally can be fed back into $[\![\mathrm{a}_1\ \text{student}]\!]^s$:

(35) $[\![\mathrm{a}_1\ \text{student}]\!]^s([\![\lambda_1\ \mathrm{a}_1\ \text{student like every}_2\ \text{teacher}]\!]^s) = 1$ iff
$[\textbf{SOME}]_1^s(\text{student})(\lambda z : \text{student}(z).\ [\textbf{EVERY}]_2^s(\text{teacher})(\lambda y.\ \text{like}(z,y)))$

We then lambda abstract again, this time over index 2:

(36) $[\![\lambda_2\ \mathrm{a}_1\ \text{student}\ \lambda_1\ \mathrm{a}_1\ \text{student like every}_2\ \text{teacher}]\!]^s$
$= \lambda x.\ [\![\mathrm{a}_1\ \text{student}\ \lambda_1\ \mathrm{a}_1\ \text{student like every}_2\ \text{teacher}]\!]^{s[2,\textsc{the}_x]}$
$= \lambda x.\ [\textsc{some}]_1^{s[2,\textsc{the}_x]}(\text{student})$
$(\lambda z : \text{student}(z).\ [\textsc{every}]_2^{s[2,\textsc{the}_x]}(\text{teacher})(\lambda y.\ \text{like}(z,y)))$
$= \lambda x.\ [\textsc{some}]_1^{s}(\text{student})(\lambda z : \text{student}(z).\ \textsc{the}_x(\text{teacher})(\lambda y.\ \text{like}(z,y)))$
$= \lambda x : \text{teacher}(x).\ [\textsc{some}]_1^{s}(\text{student})(\lambda z : \text{student}(z).\ \text{like}(z,x))$

And in our final iteration of function application, we apply $[\![\text{every}_2\ \text{teacher}]\!]^s$ to the predicate resulting from lambda abstraction:

(37) $[\![\text{every}_2\ \text{teacher}]\!]^s([\![\lambda_2\ \mathrm{a}_1\ \text{student}\ \lambda_1\ \mathrm{a}_1\ \text{student like every}_2\ \text{teacher}]\!]^s) = 1$ iff
$[\textsc{every}]_2^s(\text{teacher})$
$(\lambda x : \text{teacher}(x).\ [\textsc{some}]_1^s(\text{student})(\lambda z : \text{student}(z).\ \text{like}(z,x)))$

We have finished the derivation, but something is missing. At this point the interpretation we get is still relative to the swap state that is our parameter of interpretation: because (37) uses $[\textsc{some}]_1^s$ and $[\textsc{every}]_2^s$ instead of just **some** and **every**, the truth conditions of (37) are at the whim of s. We of course do not wish this to be the case, and instead would like to decline the opportunity to further swap. We can easily define a swap state that does precisely this: namely, STAY as defined in (38), which swaps out every etett for itself at every index:

(38) $\textsc{stay} := \lambda n \lambda D.\ D$

We then simply say that every sentence is interpreted with STAY as its swap state parameter. In that case the compositional semantics up to this point will go exactly as before—nothing in the preceding discussion relied on any particulars about the parameter s—and the final interpretation we get is as in (39):

(39) $[\![(5)]\!]^{\textsc{stay}} = 1$ iff
$\textsc{every}(\text{teacher})$
$(\lambda x : \text{teacher}(x).\ \textsc{some}(\text{student})(\lambda z : \text{student}(z).\ \text{like}(z,x)))$

One may reasonably complain that requiring the state parameter to always be STAY makes this parameter decidedly unparameterlike, as the apparent hallmark of a semantic parameter is that there is some flexibility in what that parameter can be set to. For this reason it is worth noting that I treat swap states as a parameter exclusively as a matter of formal and didactic convenience: it is an easy way to ignore swap states when they are irrelevant and use them when they are relevant. In reality, what is likely the best way to utilize swap states is to "reify" them in the semantics by including them as explicit arguments of (some) lexical items, then feeding STAY to the resulting denotation at the end of the derivation. This reification has been executed

in a variety of ways for variable assignments, so given the parallels these methods will extend equally well to swap states. In the appendix I illustrate one such implementation that adapts Charlow's (2018) approach to assignment-sensitivity, though again, many other options are possible. Regardless, in the body of the paper I will stick to parameterizing swap states as a matter of convenience.

The interpretation in (39) can be further simplified. **EVERY** and **SOME** famously meet Keenan & Stavi's (1986) condition of *conservativity*, defined in (40):

(40) D is *conservative* iff for all A and B, $D(A)(B) \Leftrightarrow D(A)(A \cap B)$

Put simply, every student smokes iff every student is a student who smokes, and some student smokes iff some student is a student who smokes. Because of this, the domain restrictions imposed by **THE** in (39)—the ones restricting x to teachers and z to students—are redundant, in that the conservativity of **EVERY** and **SOME** guarantees the same result, even absent these restrictions. Thus, (39) is equivalent to (41):

(41) $[\![(5)]\!]^{\text{STAY}} = 1$ iff $\textbf{EVERY}(\text{teacher})(\lambda x.\ \textbf{SOME}(\text{student})(\lambda z.\ \text{like}(z,x)))$

More generally, following Keenan & Stavi (1986) conservativity is often held as a universal holding of *all* determiner quantifiers, meaning that at least in terms of determiner quantification the domain restrictions imposed by **THE** make no direct semantic impact. In fact, Romoli (2015), building on work by Chierchia (1995), Fox (2002), and others, turns this convenient fact on its head, arguing that the copy theory of movement—or for our purposes, multiple-merge theories of movement more generally—could be used to *explain* the conservativity generalization. In short, Romoli's proposal is that if all quantificational DPs move at least once, then the restrictor of a lower copy will always be contained within the scope of the highest copy, and thus the scope of the quantifier always ends up being conjoined with the restrictor of that quantifier: *Det Restrictor Scope* is automatically interpreted as *Det Restrictor Restrictor+Scope*.[8] As a result, any derivation using a non-conservative quantifier will end up equivalent to the same derivation using some (often trivial) conservative quantifier. Not only is this proposal—which will be discussed in a bit more detail in Section 4.3—an interesting idea in its own right, but the observation that lower restrictors can impose conservativity will become directly relevant when we discuss

8 One may wonder why, in the semantics adopted in this paper, all quantificational DPs—and in particular object DPs—would have to undergo movement, since quantifier raising is never motivated by a type mismatch. However, there may be other motivations for movement. For example, a common approach to event semantics is to treat the denotation of the VP (or *v*P) as a predicate over events, which is then existentially closed off by a higher head such as aspect (see, e.g., Kratzer 1998, Hacquard 2006). Thus, movement may be required in order for quantificational DPs to "escape" and take scope outside of this event quantification. (However, see Champollion 2015 for an approach in which QR is not required in order for quantificational DPs to outscope event quantification.)

comparatives in the next section, as the comparative morpheme *-er* has frequently been assigned a non-conservative denotation.

Wrapping up, in this section I have offered a semantic theory that generates the effects of trace conversion in a straightforwardly bottom-up compositional semantics, without either a syntactic or a semantic operation of trace conversion. In the next section I will push this theory further, generalizing beyond DP quantification to account for modal and degree phrase scope-taking.

4 Generalizing

In this section I will show how our proposal can be extended beyond *e*-type lambda abstraction, thereby generalizing to modals and degree phrases. We start in Section 4.1 with the formal extension, revising the mechanism in a way that will permit lambda abstraction over arbitrary types. In Section 4.2 the power of this type-generalized formalism is directed toward an analysis of modals, temporarily operating under the simplifying assumption that modals come with covert, syntactically represented restrictors. This will be useful in keeping the analogy between modals' world-quantification and determiners' individual-quantification as close as possible. In Section 4.3 we will see what happens if we eschew this assumption and instead adopt the view that modals lack syntactically-represented restrictors. This relates to a broader issue, namely, what the present theory has to say about heads of some "lower" quantificational type $(\alpha t)t$, rather than $(\alpha t)(\alpha t)t$. We will see that no revisions need to be made to the analysis in order to facilitate the inclusion of such lower-type quantifiers, as we already have sufficient power to handle them. However, during this discussion we will stumble upon a way in which the theory in this paper could be exploited in order to allow hypothetical non-conservative determiners to escape Romoli's (2015) structural account of conservativity; I will offer some remarks on a potentially independently motivated grammatical constraint that might prevent such an escape. Finally, in Section 4.4 the analysis is extended to account for degree phrase scope-taking in comparatives.

4.1 Type-generalized state dependency

Let's go back to our traditional lambda abstraction, using variable assignments instead of swap states:

(23) Traditional Lambda Abstraction: (cf. Heim & Kratzer 1998)
$[\![\lambda_n \text{ X}]\!]^g = \lambda x.\ [\![\text{X}]\!]^{g[n,x]}$,
where $g[n,x]$ is the g' identical to g except that $g'(n) = x$.

Now suppose variable assignments are of type *ne*, i.e., (partial) functions from indices to individuals. Given that *e* is not the only type over which lambda abstraction takes place, a reasonable followup question is how this might be generalized in a manner that will permit lambda abstraction over arbitrary types, or at least those types over which the compositional semantics requires us to lambda abstract.

There seem to be two obvious candidate paths for type-generalizing assignment-sensitivity. The first is to utilize a single, type-flexible variable assignment: g can take an index and return an object of any (permissible) type, so $g(1)$ might be type e, $g(2)$ type d (for degrees), $g(3)$ type w (for worlds), etc. The second path is to replace a single variable assignment of type *ne* with a cluster of variable assignments for different types: one of type *ne*, one of type *nd*, etc.

Let's flesh out this second view a little more. Suppose the variable assignment g is replaced with an *assignment cluster*, a set (or tuple) containing exactly one function of type $n\alpha$ for each lambda-abstractable type α. For assignment cluster h and lambda-abstractable type α, let $h\langle\alpha\rangle$ be the $n\alpha$-type variable assignment in h, i.e., the assignment function for objects of type α. In much the same way that assignment $g[n,x]$ was the assignment g' identical to g except that $g'(n) = x$, we can define $h[n,k]$ as follows:

(42) For assignment cluster h, index n, and k of type α, $h[n,k]$ is the h' identical to h except that $h'\langle\alpha\rangle(n) = k$.

This gives us enough to define lambda abstraction over arbitrary types, as shown in (43). Notice that lambda abstractors come with not only an index, but a type parameter to indicate what type is lambda-abstracted over. This also holds of the type-generalized version of swap state lambda abstraction, meaning that the lambda-abstractors λ_1 and λ_2 in the previous section must be replaced with $\lambda_{e,1}$ and $\lambda_{e,2}$.

(43) Traditional Lambda Abstraction (Type-Generalized):
$[\![\lambda_{\alpha,n}\ \mathrm{X}]\!]^h = \lambda k_\alpha.\ [\![\mathrm{X}]\!]^{h[n,k]}$

The same general technique extends equally well to swap states. The swap states in the previous section took etetts (type $(et)(et)t$) and returned etetts. To generalize, let's say that for any type α, an *α-swap state* takes an index and an object of type $(\alpha t)(\alpha t)t$, and returns an object of type $(\alpha t)(\alpha t)t$. Thus, the swap states seen in the previous section were *e*-swap states; modals will use *w*-swap states, and degree phrases will utilize *d*-swap states. Much in the same way that we previously introduced assignment clusters, a *swap state cluster* (or *state cluster* for short) will include exactly one α-swap state for each lambda-abstractable type α, and for any state cluster r, $r\langle\alpha\rangle$ will be r's α-swap state. We can also keep the same abbreviation convention as before: for state cluster r, index n, and K of type $(\alpha t)(\alpha t)t$, $[K]^r_n := r\langle\alpha\rangle(n)(K)$. Thus, we can keep the same definitions for $[\![\mathrm{a}_n]\!]^r$ and $[\![\mathrm{every}_n]\!]^r$ as before, e.g., $[\![\mathrm{every}_n]\!]^r = [\textsc{every}]^r_n$.

In the previous section, our definition of lambda abstraction made use of the etett $\textsc{the}_x$, with x being the lambda-abstracted-over variable. In order to permit arbitrary-type lambda-abstraction, this must be generalized, so that for k of type α, $\textsc{the}_k$ will be type $(\alpha t)(\alpha t)t$. Luckily this is easy to define:

(44) For k of type α, $\textsc{the}_k$ is type $(\alpha t)(\alpha t)t$, and $\textsc{the}_k := \lambda J_{\alpha t} \lambda J'_{\alpha t} : J(k).\ J'(k)$

Finally, there is the matter of redefining $r[n, K]$ to suit our type-generalized needs. This is once again a simple matter:

(45) For state cluster r, index n, and K of type $(\alpha t)(\alpha t)t$, $r[n, K]$ is the state cluster r' that is identical to r except that for all K' of type $(\alpha t)(\alpha t)t$, $[K']^{r'}_n = K$.

We now have enough to define type-generalized lambda abstraction in our system:

(46) New Lambda Abstraction (Type-Generalized):
$[\![\lambda_{\alpha,n}\ \mathrm{X}]\!]^r = \lambda k_\alpha.\ [\![\mathrm{X}]\!]^{r[n,\textsc{the}_k]}$

Finally, recall that when we only had a single e-state s, we always evaluated relative to the state stay, essentially "declining" the chance to make any more swaps at the end of the derivation. Now that we are operating with state clusters rather than a single e-state, we will redefine stay as a state cluster, as follows:

(47) stay is the state cluster such that for all (lambda-abstractable) types α, $\textsc{stay}\langle\alpha\rangle = \lambda n \lambda K_{(\alpha t)(\alpha t)t}.\ K$.

The proposal from the previous section has now been extended in a fully type-generalized manner. We next move on to seeing how the present theory fares when it comes to modals and degree quantifiers, starting with the former.

4.2 Modals (with restrictors)

I will use the sentences in (10), repeated below, to illustrate the analysis of modal scope:

(10) a. Rivka cannot leave the party.
b. Rivka must not leave the party.

For the time being I will assume that these sentences have the LF structures in (48); note that I follow Iatridou & Zeijlstra (2013) in treating modals as merged below negation and overtly moving above it, with *must* scoping in its post-movement position and *can* reconstructing to its pre-movement position. As mentioned in the beginning of this section, I am temporarily operating under the assumption that modals have a syntactically represented restrictor res, which can be thought of as filling the role that on traditional Kratzerian accounts is also played by *if* clauses

(Kratzer 1981, 1991a,b, 2012). In Section 4.3 we will see what happens when we abandon this assumption and treat modals as restrictor-less $(wt)t$-type quantifiers. The head INT, meanwhile, serves to bring us from the realm of extensions to intensions. Thus, while *Rivka leave the party* denotes a truth value (true iff Rivka left the party in the world of evaluation), *INT Rivka leave the party* denotes a proposition true of a world iff Rivka left the party in that world.[9]

(48) a. not [can$_1$ RES] INT Rivka leave the party
b. [must$_1$ RES] $\lambda_{w,1}$ not [must$_1$ RES] INT Rivka leave the party

Our denotations for constituents will now be relative to a context parameter c and a world of evaluation parameter u. Let us put aside state clusters temporarily and assume that c and u are the only parameters of evaluation. The denotation of *Rivka leave the party*—that is, the part of the sentence below INT—is a truth value, true iff Rivka leaves the party in u:

(49) $[\![\text{Rivka leave the party}]\!]^{c,u} = 1$ iff Rivka leaves the party in u

As promised, INT then lambda-abstracts over the world of evaluation, returning a proposition, i.e., a function from worlds to truth values (type wt):

(50) $[\![\text{INT X}]\!]^{c,u} = \lambda v.\ [\![\text{X}]\!]^{c,v}$

(51) $[\![\text{INT Rivka leave the party}]\!]^{c,u} = \lambda v.$ Rivka leaves the party in v

Next up are *can$_1$* and RES. As mentioned previously, $[\![\text{RES}]\!]$ serves to restrict $[\![\text{can}]\!]$, saturating an argument that in conditionals would be saturated by the antecedent. Thus, $[\![\text{RES}]\!]^{c,u}$ will be a contextually-determined proposition R^c. This in turn makes $[\![\text{can}]\!]$ type $(wt)(wt)t$: in terms of semantic type, it is a quantifier over worlds in a manner parallel to how $[\![\text{a}]\!]$ is a quantifier over individuals. If we ignore issues of scope and copy composition, we thus might define $[\![\text{can}]\!]^{c,u}$ as in (52), where $\textsc{can}^c_u$ is a $(wt)(wt)t$-type world-quantifier that is (I) context-sensitive, allowing for differences in modal flavors; and (II) world-dependent, since what is permissible (for example) varies from world to world:

(52) $[\![\text{can}]\!]^{c,u}_{\text{preliminary}} = \textsc{can}^c_u = \lambda p \lambda q.\ \textsc{can}^c_u(p)(q)$

Note that it does not matter for our purposes what $\textsc{can}^c_u$ actually is: it might be a best-worlds quantifier à la Kratzer (1981, 1991a,b, 2012), or it might be defined in terms of probabilities and utility functions (see, e.g., Lassiter 2011). What matters for our purposes is that $\textsc{can}^c_u$ is of type $(wt)(wt)t$.

9 This is a somewhat outdated view of how possible worlds/situations enter the compositional semantics, with the more common view nowadays being that they are syntactically represented as pronouns that are bound by lambda operators (see, e.g., Percus 2000). I leave for future work the issue of how such an approach might be integrated with the analysis in this paper.

But since *can* is a modal, and at least by assumption modals can take scope via movement, we need to re-introduce state clusters into the mix: ⟦can⟧ must be state-sensitive in much the same way that ⟦a⟧ is. Since we've type-generalized our semantics by replacing swap states with state clusters, this is a simple matter:

(53) $[\![\text{can}_n]\!]^{c,u,r} = [\textsc{can}^c_u]^r_n = \lambda p \lambda q.\ [\textsc{can}^c_u]^r_n(p)(q)$

Since $\textsc{can}^c_u$ is of type $(wt)(wt)t$, $[\textsc{can}^c_u]^r_n$ will be the $(wt)(wt)t$-type world-quantifier that $r\langle w\rangle$ swaps in for $\textsc{can}^c_u$ at index n.

Let us continue with our derivation. Since $[\![\textsc{res}]\!]^{c,u,r}$ is the restrictor proposition R^c (type wt), while $[\![\text{can}_1]\!]^{c,u,r}$ is of type $(wt)(wt)t$, the former restricts the latter, leading to a $(wt)t$-type world-quantifier:

(54) $[\![\text{can}_1]\!]^{c,u,r}([\![\textsc{res}]\!]^{c,u,r}) = \lambda q.\ [\textsc{can}^c_u]^r_1(R^c)(q)$

This then composes with ⟦INT Rivka leave the party⟧^{c,u,r}, which I will abbreviate as the proposition r-leave:

(55) $[\![\text{can}_1\ \textsc{res}]\!]^{c,u,r}([\![\textsc{int}\text{ Rivka leave the party}]\!]^{c,u,r}) = 1$ iff
$[\textsc{can}^c_u]^r_1(R^c)(\text{r-leave})$

Next, this composes with ⟦not⟧, which is of type tt and simply contributes boolean negation ($[\![\text{not}]\!]^{c,u,r} = \lambda t.\ \neg t$):

(56) $[\![\text{not}]\!]^{c,u,r}([\![\text{can}_1\ \textsc{res}\ \textsc{int}\text{ Rivka leave the party}]\!]^{c,u,r}) = 1$ iff
$\neg[\textsc{can}^c_u]^r_1(R^c)(\text{r-leave})$

This gives us our final denotation relative to the state cluster r. We then evaluate relative to STAY:

(57) $[\![\text{not can}_1\ \textsc{res}\ \textsc{int}\text{ Rivka leave the party}]\!]^{c,u,\textsc{stay}} = 1$ iff $\neg\textsc{can}^c_u(R^c)(\text{r-leave})$

Of course, regardless of one's favorite theory of CAN, these will naturally be the correct truth conditions, with negation outscoping *can*: it is not permissible that Rivka leave.

We next move on to the LF in (48b), in which *must* replaces *can* and takes scope over negation:

(48b) [must$_1$ RES] $\lambda_{w,1}$ not [must$_1$ RES] INT Rivka leave the party

We can define ⟦must⟧ in a manner parallel to ⟦can⟧, but with the world-quantifier MUST replacing CAN:

(58) $[\![\text{must}_n]\!]^{c,u,r} = [\textsc{must}^c_u]^r_n = \lambda p \lambda q.\ [\textsc{must}^c_u]^r_n(p)(q)$

Up to and including negation, the derivation for (48b) runs precisely parallel to that for (48a), leading to the following result:

(59) $[\![\text{not must}_1 \textsc{res int Rivka leave the party}]\!]^{c,u,r} = 1$ iff
$\neg[\textbf{MUST}^c_u]^r_1(R^c)(\text{r-leave})$

We then lambda abstract via $\lambda_{w,1}$, following our revised rule for lambda abstraction:

(60) $[\![\lambda_{w,1}\ \text{not must}_1 \textsc{res int Rivka leave the party}]\!]^{c,u,r}$
$= \lambda v_w.\ [\![\text{not must}_1 \textsc{res int Rivka leave the party}]\!]^{c,u,r[1,\textbf{THE}_v]}$
$= \lambda v.\ \neg[\textbf{MUST}^c_u]^{r[1,\textbf{THE}_v]}_1(R^c)(\text{r-leave})$
$= \lambda v.\ \neg\textbf{THE}_v(R^c)(\text{r-leave})$
$= \lambda v : R^c(v).\ \neg\text{r-leave}(v)$

This gives us a proposition defined only for worlds in our restricted domain, and true of those worlds in which Rivka does not leave. This can naturally be fed back into $[\![\text{must}_1 \textsc{res}]\!]$:

(61) $[\![\text{must}_1 \textsc{res}]\!]^{c,u,r}([\![\lambda_{w,1}\ \text{not must}_1 \textsc{res int Rivka leave the party}]\!]^{c,u,r}) = 1$ iff
$[\textbf{MUST}^c_u]^r_1(R^c)(\lambda v : R^c(v).\ \neg\text{r-leave}(v))$

We then evaluate relative to STAY, giving us our final truth conditions:

(62) $[\![\text{must}_1 \textsc{res}\ \lambda_{w,1}\ \text{not must}_1 \textsc{res int Rivka leave the party}]\!]^{c,u,\textsc{stay}} = 1$ iff
$\textbf{MUST}^c_u(R^c)(\lambda v : R^c(v).\ \neg\text{r-leave}(v))$

Once again, we derive the correct truth conditions, with *must* scoping over negation: it is obligatory that Rivka not leave.[10]

4.3 Lower-type quantifiers, vacuous restrictors, and structural conservativity

Up to now, we have assumed that modals have a syntactically represented restrictor RES, which saturates the same argument that in conditionals is restricted by the *if* clause. However, it is not obvious that *if* clauses restrict modals through argument saturation: for example, von Fintel (1994) argues at length against this view, and in favor of an analysis in which *if* clauses effect their modal domain restriction through means other than argument saturation. If this is the case, then there is no longer any reason to assume that modals are of type $(wt)(wt)t$, with a silent head RES serving to restrict the modal. Instead, the more plausible analysis would be that modals are simply of type $(wt)t$, with no head RES at all. In this case, the LF for (10b), rather than being (48b), would instead be the simpler (63):

(48b) [must$_1$ RES] $\lambda_{w,1}$ not [must$_1$ RES] INT Rivka leave the party

(63) must$_1$ $\lambda_{w,1}$ not must$_1$ INT Rivka leave the party

10 This assumes that $\textbf{MUST}^c_u$ is a conservative quantifier over possible worlds. I am unfamiliar with any theory that makes use of non-conservative $\textbf{MUST}^c_u$, so this is presumably unproblematic.

This raises a conundrum for the present analysis. As things currently stand, all of the swap states in our state clusters trade in objects of some type $(\alpha t)(\alpha t)t$, i.e., quantifiers with restrictor arguments. If modals do not have restrictor arguments, and are thus of type $(wt)t$ instead of $(wt)(wt)t$, how do they fit into the present system? More generally, does our formalism need to be further expanded in order to account for such "lower-type" quantifiers without restrictor arguments?

Thankfully, the answer is no: we already have enough tools at our disposal to account for lower-type quantifiers. Suppose that on a traditional, copy-free semantics, the denotation for $[\![\text{must}]\!]^{c,u}$ is $\textsc{mst}^c_u$, which unlike $\textsc{must}^c_u$ takes a single propositional argument, making it type $(wt)t$. Now say that we define the $(wt)(wt)t$-type world-quantifier $\underset{\sim}{\textsc{mst}^c_u}$, which has a vacuous initial (wt)-type argument:

(64) For F of type $(\alpha t)t$, $\underset{\sim}{F}$ is type $(\alpha t)(\alpha t)t$, and $\underset{\sim}{F} := \lambda J_{\alpha t}.\, F$.

(65) $\underset{\sim}{\textsc{mst}^c_u} = \lambda p \lambda q.\, \textsc{mst}^c_u(q)$

Since $\underset{\sim}{\textsc{mst}^c_u}$ is type $(wt)(wt)t$, it is something that can be the input or output to $r\langle w\rangle$ for a given state cluster r. We can then define $[\![\text{must}_n]\!]^{c,u,r}$ as follows, with $\top$ being the vacuously true proposition (i.e., $\lambda v.\ 1$):

(66) $[\![\text{must}_n]\!]^{c,u,r} = \lambda q.\, [\underset{\sim}{\textsc{mst}^c_u}]^r_n(\top)(q)$

As desired, the semantic type for $[\![\text{must}]\!]$ is $(wt)t$ rather than $(wt)(wt)t$, but we still have something of type $(wt)(wt)t$ that is fed into the w-swap state $r\langle w\rangle$.

To see that this gets the right results, let's complete the derivation of (63). As before, the denotation up to and including the world-abstracting head INT is the wt-type proposition r-leave. This is now fed to $[\![\text{must}_1]\!]$, which is of type $(wt)t$, giving us a truth value:

(67) $[\![\text{must}_1]\!]^{c,u,r}([\![\textsc{int}\ \text{Rivka leave the party}]\!]^{c,u,r}) = 1$ iff $[\underset{\sim}{\textsc{mst}^c_u}]^r_1(\top)(\text{r-leave})$

$[\![\text{not}]\!]$ then applies, again contributing boolean negation:

(68) $[\![\text{not}]\!]^{c,u,r}([\![\text{must}_1\ \textsc{int}\ \text{Rivka leave the party}]\!]^{c,u,r}) = 1$ iff
$\neg[\underset{\sim}{\textsc{mst}^c_u}]^r_1(\top)(\text{r-leave})$

We then lambda-abstract, giving us the proposition true of those worlds in which Rivka does not leave:

(69) $[\![\lambda_{w,1}\ \text{not must}_1\ \textsc{int}\ \text{Rivka leave the party}]\!]^{c,u,r}$
$= \lambda v.\, [\![\text{not must}_1\ \textsc{int}\ \text{Rivka leave the party}]\!]^{c,u,r[1,\textsc{the}_v]}$
$= \lambda v.\, \neg[\underset{\sim}{\textsc{mst}^c_u}]^{r[1,\textsc{the}_v]}_1(\top)(\text{r-leave})$

$= \lambda v.\ \neg\textbf{THE}_v(\top)(\text{r-leave})$
$= \lambda v : \top(v).\ \neg\text{r-leave}(v)$
$= \lambda v.\ \neg\text{r-leave}(v)$

This can naturally be fed back into $[\![\text{must}_1]\!]$:

(70) $[\![\text{must}_1]\!]^{c,u,r}([\![\lambda_{w,1}$ not must$_1$ INT Rivka leave the party$]\!]^{c,u,r}) = 1$ iff $[\underset{\sim\sim}{\textbf{MST}^c_u}]^r_1(\top)(\lambda v.\ \neg\text{r-leave}(v))$

And to cap it all off, we evaluate with respect to STAY:

(71) $[\![\text{must}_1\ \lambda_{w,1}$ not must$_1$ INT Rivka leave the party$]\!]^{c,u,\text{STAY}}$
$= 1$ iff $\underset{\sim\sim}{\textbf{MST}^c_u}(\top)(\lambda v.\ \neg\text{r-leave}(v))$
$= 1$ iff $\textbf{MST}^c_u(\lambda v.\ \neg\text{r-leave}(v))$

In summary, then, the present proposal extends equally well to lower-type quantifiers: in this case, $[\![\text{must}]\!]$ was type $(wt)t$, rather than $(wt)(wt)t$, but swap states were equally effective in allowing for direct composition.

This formal technique raises interesting questions about whether and how multiple-merge theories of movement can be used to account for the conservativity hypothesis—the hypothesis that all quantificational determiners have conservative denotations—as discussed in detail by Romoli (2015). To see why, let us first go over the rough details of a Romoli-style account. Consider the etett **SOMENON**, defined in (72):

(72) $\textbf{SOMENON} := \lambda P \lambda P'.\ \exists x[\neg P(x) \wedge P'(x)]$

SOMENON is non-conservative. $\textbf{SOMENON}(A)(B)$ is true iff some non-A is a B, while $\textbf{SOMENON}(A)(A \cap B)$ is true if some non-A is both an A and a B. The former could clearly be a contingent proposition, while the latter is necessarily contradictory. Thus, $\textbf{SOMENON}(A)(B)$ is not necessarily equivalent to $\textbf{SOMENON}(A)(A \cap B)$, and **SOMENON** is non-conservative.

Now suppose we have a determiner *somenon*, which is defined based on **SOMENON** in a manner parallel to how $[\![\text{a}]\!]$ and $[\![\text{every}]\!]$ are defined based on **SOME** and **EVERY**, respectively:

(73) $[\![\text{somenon}_n]\!]^r = [\textbf{SOMENON}]^r_n$

Suppose in addition that we have the LF in (74) for the sentence *Somenon student left*:

(74) [somenon$_1$ student] $\lambda_{e,1}$ T [somenon$_1$ student] left

We will see that as desired, when we semantically compose (74), a (vacuous) conservative interpretation is "forced" by the lower copy: the restrictor of the lower copy is in the scope of the higher copy, so $\textbf{SOMENON}(A)(B)$ is interpreted as $\textbf{SOMENON}(A)(A \cap$

B). Since such an interpretation is necessarily contradictory, as discussed above, we will end up with a vacuously false interpretation. We start with the VP, whose composition is relatively straightforward:

(75) $[\![\text{somenon}_1 \text{ student left}]\!]^r = 1$ iff $[\textbf{SOMENON}]^r_1(\text{student})(\text{left})$

Assuming again that tense (T) is vacuous, we next move on to lambda abstraction via $\lambda_{e,1}$:

(76) $[\![\lambda_{e,1} \text{ (T) somenon}_1 \text{ student left}]\!]^r$
$= \lambda x.\ [\![\text{(T) somenon}_1 \text{ student left}]\!]^{r[1,\textbf{THE}_x]}$
$= \lambda x.\ [\textbf{SOMENON}]^{r[1,\textbf{THE}_x]}_1(\text{student})(\text{left})$
$= \lambda x.\ \textbf{THE}_x(\text{student})(\text{left})$
$= \lambda x : \text{student}(x).\ \text{left}(x)$

And in the final step, this re-composes with *somenon student*. The result of this computation, evaluated with respect to STAY, can be seen in (77):

(77) $[\![\text{somenon}_1 \text{ student}]\!]^{\text{STAY}}([\![\lambda_{e,1} \text{ (T) somenon}_1 \text{ student left}]\!]^{\text{STAY}})$
$= 1$ iff $\textbf{SOMENON}(\text{student})(\lambda x : \text{student}(x).\ \text{left}(x))$
$= 1$ iff $\exists y[\neg\text{student}(y) \wedge [\lambda x : \text{student}(x).\ \text{left}(x)](y)]$

The first conjunct inside the existential quantification requires that bound y is not a student, while the second requires that a predicate, restricted to students, holds of y. This is contradictory, meaning that there can be no such y: the denotation in (77) is false by logical necessity, as desired.

However, there is a way of defining $[\![\text{somenon}]\!]$ based on **SOMENON** that is compatible with the general theory offered in this paper, but that also avoids the contradictory reading generated by the interpretation in (73). Naturally, **SOMENON** is an etett, and thus of type $(et)(et)t$, meaning that for any et-type predicate P, $\textbf{SOMENON}(P)$ is type $(et)t$. But this in turn means that $\textbf{SOMENON}(P)$, which tacks on a vacuous et-type argument, is itself an etett that can be swapped in or out via swap state. Now suppose that we instead define $[\![\text{somenon}]\!]$ as in (78):[11]

(78) $[\![\text{somenon}_n]\!]^r = \lambda P \lambda P'.\ [\textbf{SOMENON}(P)]^r_n(\top)(P')$

When we compose the VP, we get the following result:

(79) $[\![\text{somenon}_1 \text{ student left}]\!]^r = 1$ iff $[\textbf{SOMENON}(\text{student})]^r_1(\top)(\text{left})$

Next, we lambda abstract. Notice that this time, what is swapped out for **THE** is not **SOMENON**, but $\textbf{SOMENON}(\text{student})$:

11 Naturally, $\top$ here is the vacuous et- (rather than wt-)type predicate. The notation is left ambiguous for simplicity, since the type can be gleaned from context.

(80) $[\![\lambda_{e,1}\ (\mathrm{T})\ \text{somenon}_1\ \text{student left}]\!]^r$
$= \lambda x.\ [\![(\mathrm{T})\ \text{somenon}_1\ \text{student left}]\!]^{r[1,\textsc{the}_x]}$
$= \lambda x.\ [\underbrace{\textsc{somenon}(\text{student})}]_1^{r[1,\textsc{the}_x]}(\top)(\text{left})$
$= \lambda x.\ \textsc{the}_x(\top)(\text{left})$
$= \lambda x : \top(x).\ \text{left}(x)$
$= \lambda x.\ \text{left}(x)$

This naturally recomposes with $[\![\text{somenon}_1\ \text{student}]\!]$; the result, when evaluated with respect to STAY, is as follows:

(81) $[\![\text{somenon}_1\ \text{student}\ \lambda_{1,e}\ (\mathrm{T})\ \text{somenon}_1\ \text{student left}]\!]^{\textsc{stay}}$
$= 1$ iff $\underbrace{\textsc{somenon}(\text{student})}(\top)(\text{left})$
$= 1$ iff $\textsc{somenon}(\text{student})(\text{left})$

We thus end up with a perfectly fine, non-contradictory interpretation: some non-student left. Therefore, if we wish to integrate the present approach with an analysis according to which multiple-merge theories of movement account for the conservativity universal, something needs to be said about why a denotation like (78) is not a permissible determiner denotation.

Formally speaking, it is easy to construct a requirement that prevents a lexical quantifier denotation of the sort seen in (78):

(82) For every determiner D, there is an etett D such that $[\![\mathrm{D}_n]\!]^r = [D]^r_n$.

This blocks the gimmick seen in (78), and any determiner interpretation whose quantificational force is **SOMENON** must have a denotation along the lines of (73), which as we saw above leads to a contradictory interpretation. But the more interesting question is, why should the constraint in (82) hold in the first place?

While I lack a definitive answer, I will sketch out one possibility. Notice that in the derivation using the definition of *somenon* in (78), the semantic contribution of the restrictor of the lower copy of the DP was completely wiped out by lambda abstraction, as seen in (80). In other words, if $[\![\text{student}]\!]$ were replaced with any other *et*-type predicate at the lower copy, the resulting interpretation would be exactly the same. We can thus refer to the lower copy of *student* as *semantically erased*; this is more formally defined in (83):

(83) If constituent X is semantically contentful (i.e., in the domain of $[\![\cdot]\!]$) and contained in constituent Y at LF, *X is semantically erased in Y* iff for every J, replacing $[\![\mathrm{X}]\!]$ with J in the semantic derivation leads to an interpretation that is either undefined or identical to $[\![\mathrm{Y}]\!]$.[12]

12 The "undefined" disjunct covers those J that simply cannot be inserted in place of $[\![\mathrm{X}]\!]$, e.g., if they are of the wrong type.

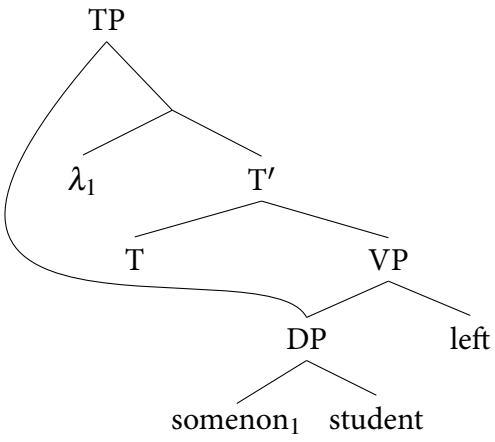

Figure 3 Multidominance LF for *Somenon student left*.

We can then define what I will call the *Anti-Erasure Principle*, inspired by Chomsky's (1995) *Full Interpretation* and Gajewski's (2002) *L-analyticity*, which simply states that any LF representation is ill-formed if it contains a semantically erased constituent:

(84) Anti-Erasure Principle: An LF structure Y is ill-formed if it contains a subconstituent X that is semantically erased in Y.

The Anti-Erasure Principle would not militate against a lexical denotation like (78) *per se*. But any derivation in which such a DP takes scope at a position other than where it is merged will be blocked, since the restrictor of the lower copy will necessarily be semantically erased.

Notice that for the current formulation of the Anti-Erasure Principle to work, each copy of a given constituent must be counted separately. After all, cumulatively, the two copies of *student* are not semantically erased, since the higher copy contributes the restriction of *somenon*; it is only the lower copy of *student* on its own that is semantically erased. This in turn means that if we switch from copy theory to multidominance theory, the Anti-Erasure Principle needs to be tweaked slightly. To see why, consider the LF in Fig. 3, which is the same LF as (74) with copies replaced by multidominance structures. Naturally, this multidominance structure would semantically compose in a fashion identical to its copy-theoretic counterpart. But since there is only one "copy" of *student*, we can no longer say that there is a lower, semantically erased copy and a higher, unerased copy. Instead, if we replace ⟦student⟧ in this structure with ⟦teacher⟧, for example, we will end up with a distinct (and well-defined) denotation, meaning that *student* is not semantically erased.

This can be fixed by strengthening the Anti-Erasure Principle as follows:

(85) Anti-Erasure Principle (Revised): An LF structure Z is ill-formed if it contains constituents Y and X such that X is a subconstituent of Y, and X is semantically erased in Y.

To see that this fixes the problem for multidominance structures, consider again the LF in Fig. 3, which plays the role of Z in the definition in (85). Naturally, *student* will play the role of X. As for Y, consider the constituent that includes the lambda abstractor λ_1 and T′, excluding the higher landing site of *somenon student*. Since *student* is indeed semantically erased in this constituent—the denotation, where defined, will always be $\lambda x.\ \mathrm{left}(x)$—by the finalized Anti-Erasure Principle the structure in Fig. 3 is illicit. Since the new Anti-Erasure Principle is strictly stronger than the old one, the copy structure in (74) remains disallowed.[13]

As a final note, observe that the Anti-Erasure Principle does not cause any problems for, say, our current definitions of $[\![\mathrm{a}_n]\!]$ and $[\![\mathrm{every}_n]\!]$. This may not be obvious, since lambda abstraction does "erase" the quantificational force of lower copies of these determiners by replacing them with bound **the**: lambda abstraction over index 1 gets the same result if a lower copy is interpreted as $[\textbf{every}]^r_1$ as it does if it is interpreted as $[\textbf{some}]^r_1$, since in either case this is replaced with **the**. But to see that there is no problem, consider the LF for *A student left*:

(86) $[\mathrm{a}_1$ student$]$ λ_1 (T) $[\mathrm{a}_1$ student$]$ left

Now consider the constituent up to and including lambda abstraction. The denotation for this constituent is $\lambda x : \mathrm{student}(x).\ \mathrm{left}(x)$. One may think from this that the determiner *a* has been semantically erased in this constituent, since its quantificational force is gone. But if it were truly semantically erased, we would expect that for any etett D, replacing $[\![\mathrm{a}_1]\!]^r$ with D in the derivation should lead to either an undefined interpretation or the same interpretation. This is not the case. As an extreme example, consider the vacuous $D_\top$:

(87) $D_\top := \lambda P \lambda P'.\ 1$

When $D_\top$ replaces $[\![\mathrm{a}_1]\!]^r$, because of the vacuity of $D_\top$ the denotation prior to lambda abstraction is simply the truth value 1. Lambda abstraction then tacks on a vacuous

13 This strengthening of the Anti-Erasure Principle may be motivated even within the confines of the copy theory of movement. As mentioned above, the original formulation required that co-copies be treated separately, since one copy of *student* was semantically erased, and the other was not. But in other areas of syntax, and in particular the interface between syntax and phonology, the opposite conclusion has been reached: namely, that the syntactic apparatus treats co-copies as a *singular* syntactic object. For example, Nunes (2004) adopts this view in his account of why only one copy in a given chain is (usually) pronounced. It thus may be desirable to posit the same principle at LF: the Anti-Erasure Principle cannot count copies separately and must treat chains "cumulatively", meaning that copy structures are treated just like multidominance structures.

entity argument, meaning that the interpretation after lambda abstraction is the vacuously true predicate $\lambda x.\ 1$. Since $\lambda x.\ 1$ is a defined and distinct denotation, quantificational determiners are not semantically erased by lambda abstraction, unlike ⟦student⟧ in the illicit derivation with *somenon*.

To summarize, by utilizing a formal mechanism that introduces a vacuous restrictor argument, then using the vacuously true predicate as the restrictor of the resulting quantifier, the present system can account for $(\alpha t)t$-type quantifiers. However, if left unconstrained, this mechanism could be used to circumnavigate a Romoli (2015)-style account of the conservativity universal. While I cannot offer a complete rebuttal in this paper, one possibility is that such uses are prevented by the Anti-Erasure Principle, an intuitively plausible principle that disallows any substructure in which meaningful constituents make no semantic contribution.

In wrapping up this section, we next turn to degree phrases in comparatives.

4.4 Degree phrases

4.4.1 The problem of non-conservative *-er*

Before offering an analysis of degree phrases, it will help to discuss classical accounts of comparatives in the tradition of von Stechow (1984) and Heim (1985, 2000). We will use (88) as our sample sentence:

(88) Jo is taller than Al is.

On traditional accounts (88) has an LF along the lines of (89), in which the degree phrase headed by *-er* undergoes QR:

(89) [-er$_1$ Op$_2$ λ_2 than Al is tall t_2] λ_1 Jo is tall t_1

⟦tall⟧ is a relation between a degree d and individual x, true iff x is at least d-tall:

(90) $[\![\text{tall}]\!]_{\text{traditional}} = \lambda d \lambda x.\ \text{height}(x) \geq d$

Op is a *wh*-like operator that triggers lambda abstraction over elided *tall*'s degree argument, while *than* is generally treated as semantically vacuous. Therefore, the denotation of the restrictor of *-er* is a degree predicate true of d iff Al is at least d-tall, and the denotation of the scope of *-er* is a degree predicate true of d iff Jo is at least d-tall. ⟦-er⟧ is thus of type $(dt)(dt)t$, i.e., a degree quantifier.

(91) $[\![(88)]\!]_{\text{traditional}} = [\![\text{-er}]\!](\lambda d.\ \text{height}(\text{al}) \geq d)(\lambda d.\ \text{height}(\text{jo}) \geq d)$

Two common denotations for *-er* that generate the correct truth conditions for (88) are provided in (92).

(92) a. $[\![\text{-er}]\!]_{\text{take 1}} = \lambda D \lambda D'.\ \max(D) < \max(D')$

b. $[\![\text{-er}]\!]_{\text{take 2}} = \lambda D \lambda D'.\ \exists d[\neg D(d) \wedge D'(d)]$

By using (92a), the predicted truth conditions are that the maximal degree not exceeding Al's height—that is, Al's height itself—is less than the maximal degree not exceeding Jo's height. In other words, Jo's height exceeds Al's.

(93) $[\![(88)]\!]_{\text{take 1}}$
$= 1$ iff $\max(\lambda d.\ \text{height}(\text{al}) \geq d) < \max(\lambda d.\ \text{height}(\text{jo}) \geq d)$
$= 1$ iff $\text{height}(\text{al}) < \text{height}(\text{jo})$

(92b), meanwhile, is a degree version of **SOMENON**; let's call it **D-SOMENON**. Using **D-SOMENON** in the LF in (89) leads to the following truth conditions: there is a degree that is not less than or equal to Al's height, and that is less than or equal to Jo's height. This will only be the case if Jo's height exceeds Al's.

(94) $[\![(88)]\!]_{\text{take 2}}$
$= 1$ iff $\textbf{D-SOMENON}(\lambda d.\ \text{height}(\text{al}) \geq d)(\lambda d.\ \text{height}(\text{jo}) \geq d)$
$= 1$ iff $\exists d[\text{height}(\text{al}) \ngeq d \wedge \text{height}(\text{jo}) \geq d]$

Importantly, both definitions in (92) are non-conservative. Clearly this is true of (92b) (**D-SOMENON**), for the same reason as with **SOMENON**. Meanwhile, if (92a) were conservative, we would expect it to be equivalent to (95):

(95) $\lambda D \lambda D'.\ \max(D) < \max(D \cap D')$

But since D is necessarily a superset of $D \cap D'$, it is impossible for the maximal degree in the latter to exceed the maximal degree in the former. Thus, (95) is not equivalent to (92a)—in fact, it's contradictory—so (92a) is non-conservative.

Since degree phrases take scope by means of QR, the non-conservativity of (92a) and (92b) presents a *prima facie* problem for the most straightforward multiple-merge version of the traditional analysis, illustrated informally in (96):

(96) [-er$_1$ than Al is tall] λ_1 Jo is tall [-er$_1$ than Al is tall]

Importantly, the restrictor of *-er* (the *than* clause) makes its semantic contribution at both the lower and higher merge sites. As discussed above in reference to Romoli's (2015) structural account of conservativity, this means that conservativity is essentially imposed: the swapping-in of **THE** at the lower merge site ensures that the scope of the degree phrase will be restricted to degrees not exceeding Al's height. As a result, if **D-SOMENON** is used, then the denotation of (88) is predicted to be (97), which is contradictory for the same reason as (98), taken from (77):

(97) $\textbf{D-SOMENON}(\lambda d.\ \text{height}(\text{al}) \geq d)(\lambda d : \text{height}(\text{al}) \geq d.\ \text{height}(\text{jo}) \geq d)$

(98) $\textbf{SOMENON}(\text{student})(\lambda x : \text{student}(x).\ \text{left}(x))$

The same goes for using (92a), which should lead to the contradictory (99):

(99) $\max(\lambda d.\ \text{height}(\text{al}) \geq d) < \max(\lambda d : \text{height}(\text{al}) \geq d.\ \text{height}(\text{jo}) \geq d)$

So how do we avoid this? One possibility is that we exploit the conservativity-dodging trick introduced in Section 4.3. For example, $[\![\text{-er}]\!]^r$, rather than being the more straightforward (100a), may instead be the somewhat more roundabout (100b), parallel to the hypothetical denotation for *somenon* in (78):

(100) a. $[\![\text{-er}_n]\!]^r \stackrel{?}{=} [\textsc{d-somenon}]^r_n$

b. $[\![\text{-er}_n]\!]^r \stackrel{?}{=} \lambda D \lambda D'.\ [\textsc{d-somenon}(D)]^r_n(\top)(D')$

Recall that for hypothetical *somenon*, this sort of denotation dodged forced conservativity by exploiting semantic erasure: the restrictor's contribution at the lower merge site was semantically erased by lambda abstraction. The same thing would happen here, leading to the interpretation in (101) (= (94)) instead of (97).

(101) $\textsc{d-somenon}(\lambda d.\ \text{height}(\text{al}) \geq d)(\lambda d.\ \text{height}(\text{jo}) \geq d)$ = (94)

This gives us the correct, non-contradictory truth conditions: there is a degree of height that is not less than or equal to Al's height, but that is less than or equal to Jo's height. But in doing this we also pull the rug out from under our Romoli (2015)-style structural account of the conservativity of DP quantification: why is a denotation like (100b) allowed for *-er*, but not for hypothetical *somenon*? For example, this approach would necessitate the abandonment of the Anti-Erasure Principle, which would be violated here in much the same way as for *somenon*. While structural accounts of DP conservativity may indeed have to be jettisoned, it is worth seeing if there is some other way in which we can get out of our present predicament.

If we wish to avoid the problem of forced conservativity without abandoning the Anti-Erasure Principle, two possibilities immediately present themselves. One is that the degree quantifier itself is in fact conservative, with something else giving the illusion of a non-conservative quantifier. The second is that the restrictor of *-er* does not appear or is uninterpreted at the lower merge site, so that a conservative interpretation is not actually forced upon us.

The idea that the degree quantifier in comparatives is actually conservative has been proposed on independent grounds by Schwarzschild (2008) and Gajewski (2008), who in our terminology operate within a **D-SOMENON**-based semantics for comparatives, but who decompose **D-SOMENON** into a "plain" existential degree quantifier and a syntactic negation in the *than* clause. Thus, rather than having a non-conservative degree quantifier **D-SOMENON** with restrictor $\lambda d.\ \text{height}(\text{al}) \geq d$, we end up with a conservative degree quantifier **D-SOME** with the negated restrictor $\lambda d.\ \text{height}(\text{al}) \ngeq d$:

(102) $\textsc{d-some}(\lambda d.\ \text{height}(\text{al}) \ngeq d)(\lambda d.\ \text{height}(\text{jo}) \geq d)$

This is, of course, equivalent to (101). But there's an important difference: this time, if the restrictor of **D-SOME** also restricts the scope of **D-SOME**, as in (103), we no longer get a contradictory interpretation like (98); in fact, thanks to the conservativity of **D-SOME**, the result is equivalent to (102).

(103) $\textbf{D-SOME}(\lambda d.\ \text{height}(\text{al}) \not\geq d)(\lambda d : \text{height}(\text{al}) \not\geq d.\ \text{height}(\text{jo}) \geq d)$

Decomposing *-er* in this manner thus avoids the problem of forced conservativity, since the degree quantifier is already conservative to begin with.

In the rest of this section I will explore the second possibility: the *than* clause is somehow absent, either syntactically or semantically (or both), from lower merge sites of *-er*. In particular, I will adopt a slightly revised version of Bhatt & Pancheva's (2004) analysis, in which *-er* in fact denotes a non-conservative degree quantifier, but the *than* phrase is not merged into the structure until after *-er* has reached its final landing spot (*late merge*). We will see that this avoids the problem of forced conservativity: the restrictor makes no contribution at lower merge sites, so forced conservativity does not arise.

4.4.2 Late merging the *than* clause

One of the guiding observations made by Bhatt & Pancheva (2004) is that while *than* clauses are interpreted as the complement of *-er*, they (almost) always have to undergo extraposition, i.e., the same syntactic process as that separating the relative clause from *book* in (104).

(104) Rivka read [every book] yesterday [that Kwame suggested to her].

In cases like (88) this extraposition is seemingly string-vacuous, but in nominal comparatives this is no longer the case.[14]

(105) a. * Jo bought more than Al did cars.
b. Jo bought more cars than Al did.

If we adopt the common view that *more* is *much*/*many* + *-er* (Bresnan 1973 and much work since), then given that the *than* clause is the sister of *-er* the ill-formed (105a) should be the result of a derivation without *than* clause extraposition. Once the *than* clause is extraposed we get the well-formed (105b).[15]

14 An exception to this rule noted by Bhatt & Pancheva (2004) is when the complement of *than* is a cardinality-denoting phrase, as in *more than three books* (cf. *?more books than three*). This extends to degree-denoting phrases, e.g., *more than three pounds of rice* (cf. *?more (*of) rice than three pounds*). See their fn. 31 (p. 40) for a few observations about this puzzle.

15 Note that this is not tied to the ellipsis in the *than* clause: we see the same thing in nominal comparative subdeletion constructions, as evidenced by the contrast between *Jo bought more cars than Al bought boats* and **Jo bought more than Al bought boats cars*.

(106) a. * Jo bought many [-er than Al did] cars.
b. Jo bought many [-er] cars [than Al did].

Bhatt & Pancheva (2004) provide robust evidence that *than* clauses are indeed extraposed, based on a variety of parallels between *than* clauses and extraposed relative clauses of the sort seen in (104). They then adopt the analysis of extraposition proposed by Fox & Nissenbaum (1999) and Fox (2002), in which QR has a pivotal role to play. In short, Fox & Nissenbaum follow Lebeaux (1990) in positing that adjoined constituents like relative clauses can be *late merged*, i.e., countercyclically adjoined to the constituent they modify after that constituent has already merged into a larger structure and undergone movement. Extraposition is the result of late-merging the relative clause after QR has taken place; while only the lower copy of the determiner and its nominal complement is pronounced, the relative clause is merged with the higher (unpronounced) copy of the NP and pronounced there.[16] If QR is stipulated to be a rightward operation, this generates the correct word order:

(107) Rivka read [every book] yesterday [every book that Kwame suggested to her]

Among other motivations, this accounts for what they call Williams's Generalization (after Williams (1974)), which states that a DP must scope at least as high as anything extraposed from it. An illustration of Williams's Generalization can be seen in (108):

(108) Illustration of Williams's Generalization (Fox 2002: 72):
a. I read every book that John had recommended before you did.
b. I read every book before you did that John had recommended.

(108a) is ambiguous between two readings. If *every* scopes below *before*, the resulting interpretation is that I was the first to make my way through John's whole list, though certain books you might have finished before me. If *every* scopes above *before*, the interpretation is stronger: each book in the list was finished by me first. But (108b), in which the relative clause is extraposed past *before*, is unambiguous, and only the latter reading is available. This is precisely what is expected if extraposition is tied to QR in the manner proposed by Fox & Nissenbaum, since *every book* must QR past *before you did* in order for the relative clause to be late merged where it is.

With this in mind, Bhatt & Pancheva (2004) propose that in comparatives the *than* clause is late merged with *-er*, so that only *-er* is merged in the lower position. In (109) this is roughly illustrated for the nominal comparative in (105b):

(109) Jo bought many [-er] cars [-er than Al did]

16 While I frame this discussion in copy-theoretic terms, late merge is equally possible in multidominance theories of movement.

So how are such structures interpreted? Let us take (88) as our example. In this case, the derived LF will look like (110a). Note that (I) QR here is rightward, meaning that the scope of *-er* is to its left; and (II) as stated previously, extraposition here is string-vacuous, since no overt material intervenes between where *-er* is pronounced and where the extraposed *than* clause ends up. After inserting lambda abstraction nodes and performing trace conversion on lower copies of *-er*, the final interpreted LF is as in (110b).

(110) a. $\underbrace{[\text{Jo is tall } [\text{-er}_1]]}_{\text{scope}}$ $[\text{-er}_1$ $\underbrace{\text{than Al is tall } [\text{-er}_1]}_{\text{restrictor}}]$

b. $\underbrace{[\lambda d_1 \text{ Jo is tall the } d_1]}_{\text{scope}}$ $[\text{-er}_1$ $\underbrace{\lambda d_1 \text{ than Al is tall the } d_1}_{\text{restrictor}}]$

Assuming either of the traditional non-conservative denotations for *-er* in (92), the LF in (110b) leads to the same correct interpretation as the more traditional LF in (89), repeated below.

(89) $[\text{-er}_1\ \text{Op}_2\ \lambda_2$ than Al is tall $t_2]\ \lambda_1$ Jo is tall t_1

Importantly, Bhatt & Pancheva's (2004) proposal avoids the problem of forced conservativity while retaining a non-conservative semantics for *-er*. This is because the restrictor of *-er* (the *than* clause) is not merged until after the degree phrase has reached its landing site, meaning that it does not restrict the scope of the degree phrase: just like on the traditional approach, the scope is just $\lambda d.\ \text{height}(\text{jo}) \geq d$. This can be translated directly into the analysis offered in this paper: we can keep a non-conservative definition of *-er*, so long as the *than* clause is late merged.

Before executing this translation into the present semantic framework, two revisions to Bhatt & Pancheva's (2004) syntactic analysis are required. The first is obvious: the whole point of the proposal in this paper is that we do not utilize a syntactic operation of trace conversion, meaning that at all points in the structure the degree head will be *-er*, not *the*. As for the second revision, notice that in the pre-trace-conversion LF in (110a), the lower copies of *-er* have no complement, meaning that they have no restrictor. For Bhatt & Pancheva it is okay for *-er* to not have a restrictor at lower merge sites, since their version of trace conversion inserts one. But for us this is not an option, and the composition breaks down if any copy of *-er* does not have its first argument saturated. In other words, while *-er* cannot have the *than* clause as its restrictor at lower merge sites, it does have to be restricted by *something*.

There are a few ways this could be resolved. The first is to stipulate that a silent head H, whose denotation is the vacuously true degree predicate ($\top$), is inserted as the complement of *-er* at lower merge sites. In this case, the LF structure for (86) will be as in Fig. 3. However, this doesn't seem like a desirable choice. After all, one of the core arguments against a syntactic operation of trace conversion is that it requires

the countercyclic insertion of semantically interpreted lexical material, in violation of the otherwise robust Inclusiveness Condition. An analysis in which syntactic trace conversion is avoided, but in which some other semantically interpreted head must be inserted for separate reasons, does not actually solve this problem, but instead simply shifts it elsewhere.

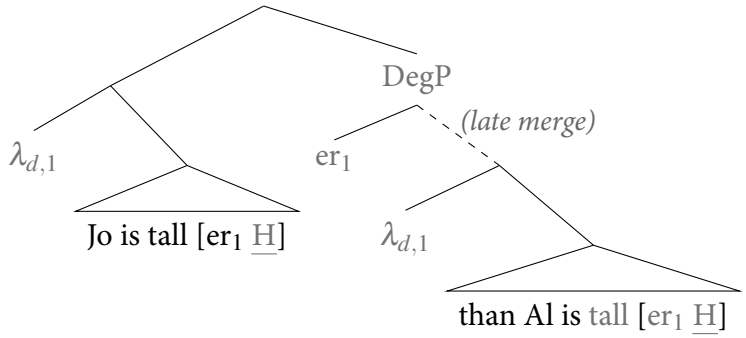

Figure 4 First potential LF structure for (88). (underlined = inserted)

A second possibility is to avoid H-insertion by adding a semantic rule that performs the equivalent of H-insertion in cases where a phrase-level constituent has a restrictor-less denotation:

(111) Restriction Insertion (RI):
If $[\![\text{XP}]\!]^r$ is of type $(\alpha t)(\alpha t)t$, then $[\![\text{XP Y}]\!]^r$ is the result of composing $[\![\text{Y}]\!]^r$ with $[\![\text{XP}]\!]^r(\lambda k_\alpha.\ 1)$.

In this case we could more or less keep Bhatt & Pancheva's (2004) pre-trace conversion LF in (110a), since RI will apply to restrictor-less *-er*, which is a DegP on its own. Translated into our system, this LF will look like Fig. 5.

A third possibility, which I will adopt in the rest of this paper, starts like the first in that there is a head H whose denotation is the vacuous degree predicate $\top$. However, rather than stipulating that H is LF-inserted as the complement of *-er* at lower merge sites, we could instead say that H is the complement of *-er* all along, thus appearing at both higher and lower merge sites. At lower merge sites H restricts *-er* on its own, while at the higher merge site the *than* clause is late merged as the complement of H, as illustrated in Fig. 5. As a result, at lower merge sites *-er* is restricted by vacuous H. Meanwhile, at the higher merge site, H and the *than* clause compose together. Since each denotes a degree predicate, the two compose via Predicate Modification, i.e., intersection. But since $[\![\text{H}]\!]$ is the vacuously true degree predicate, intersecting this with the denotation of the *than* clause simply gives us the latter back. Hence, we

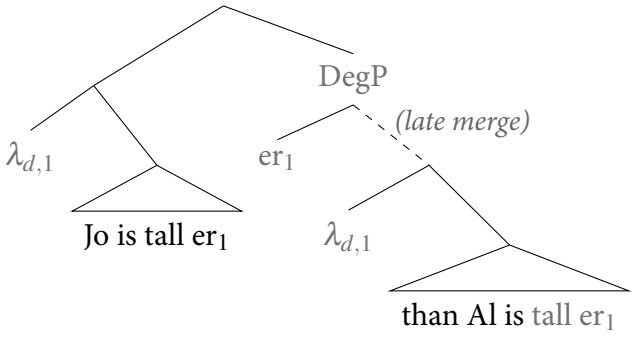

Figure 5 Second potential LF structure for (88).

get vacuous restriction at lower merge sites, and restriction by the *than* clause at the highest merge site, as desired.

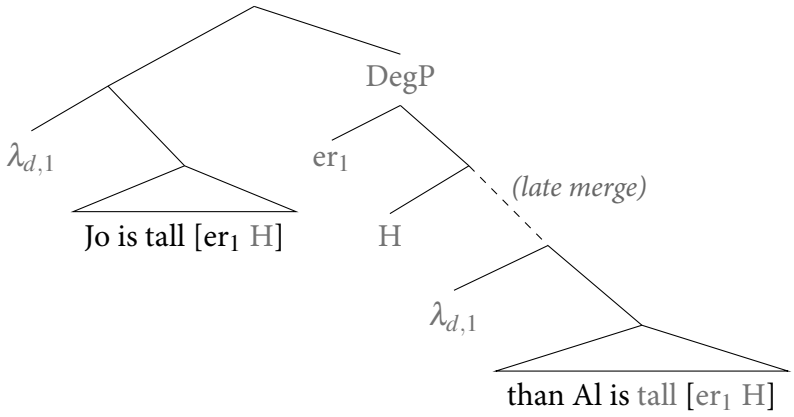

Figure 6 Third potential LF structure for (88).

This approach avoids the Inclusiveness-violating stipulation of H-insertion, as well as the semantic stipulation of Restriction Insertion. That being said, Bhatt & Pancheva (2004) argue specifically for the claim that the *than* phrase is a complement of *-er*, based primarily on the observation that there are selectional restrictions between the degree head and the head of its clausal complement:

(112) a. Jo is taller {than/*as} Al is.
b. Jo is as tall {as/*than} Al is.

If *than* (or *as*) heads the complement of *-er*, these selectional requirements can of course be explained quite straightforwardly, in contrast to if, say, the *than* phrase

was some form of adjunct. But this explanation works just as well for the present approach. Suppose that there are two versions of H: H_{than} and H_{as}, both of which denote the vacuous degree predicate. Suppose in addition that the former selects for a *than* phrase as its complement, while the latter selects for an *as* phrase. If *-er* selects for H_{than}, while *as* selects for H_{as}, then the selectional restrictions are enforced without the *than* phrase being the complement of *-er*: all that matters is that it be the complement of H and that H head the complement of *-er*. In other words, as long as the "thread" of head-complement relations is unbroken, selectional restrictions can be imposed, regardless of whether the *than* phrase is the complement of *-er* or the complement *of the complement* of *-er*.[17]

Adopting the structure in Fig. 6 as our LF for (88), we next turn to the task of semantic composition, starting with the *than* clause. With respect to *-er*, for our purposes there is no difference between using a denotation based on **D-SOMENON** and using one based on max, so for concreteness's sake I will use the former:

(113) $[\![\text{-er}_n]\!]^r = [\textbf{D-SOMENON}]^r_n = \lambda D \lambda D'.\ [\textbf{D-SOMENON}]^r_n(D)(D')$

Naturally, $[\![\text{-er}_1]\!]$ composes with $[\![\text{H}]\!]$:

(114) $[\![\text{-er}_1]\!]^r([\![\text{H}]\!]^r) = \lambda D'.\ [\textbf{D-SOMENON}]^r_1(\top)(D')$

This next composes with $[\![\text{tall}]\!]$. As mentioned before, a traditional analysis would assign *tall* the denotation in (90), true of d and x iff x is at least d-tall.

(90) $[\![\text{tall}]\!]_{\text{traditional}} = \lambda d \lambda x.\ \text{height}(x) \geq d$

However, in much the same way that the verb *like* had to be assigned a higher type to allow composition with quantificational DP arguments, now *tall* must be assigned a higher type to allow composition with quantificational DegP arguments, i.e., arguments of type $(dt)t$ instead of d. This higher-type version can be seen in (115).

(115) $[\![\text{tall}]\!]^r = \lambda G_{(dt)t} \lambda x.\ G(\lambda d.\ \text{height}(x) \geq d)$

$[\![\text{tall}]\!]^r$ can now compose with $[\![\text{-er}_1\ \text{H}]\!]^r$:

(116) $[\![\text{tall}]\!]^r([\![\text{-er}_1\ \text{H}]\!]^r)$
$= \lambda x.\ [\![\text{-er}_1\ \text{H}]\!]^r(\lambda d.\ \text{height}(x) \geq d)$
$= \lambda x.\ [\textbf{D-SOMENON}]^r_1(\top)(\lambda d.\ \text{height}(x) \geq d)$

17 Lebeaux (1990) argues that only adjuncts can be late merged, and not complements, based on a variety of differences between adjunct and complement relative clauses. Bhatt & Pancheva (2004) argue that the apparent adjunct-complement distinction can be accounted for through semantic means: in short, the composition falls apart when trying to late merge an argument relative clause. Their proposal extends equally well to my own analysis, but there is also another possibility: perhaps the distinction is not between syntactic arguments and adjuncts, but rather between semantic arguments and intersective modifiers. On my analysis, *than* phrases are *syntactic* arguments of H, but *semantically* they function as intersective modifiers, like adjunct relative clauses and unlike argument relative clauses. I leave further exploration of this possibility for future work.

We next combine with $[\![\text{Al}]\!]$, which I take to be the individual al.[18] This gets us (117):

(117) $[\![\text{tall er}_1\ \text{H}]\!]^r([\![\text{Al}]\!]^r) = 1$ iff $[\textsc{d-somenon}]^r_1(\top)(\lambda d.\ \text{height}(\text{al}) \geq d)$

Sticking with our assumption that *than* is semantically vacuous, the next step is lambda abstraction:

(118) $[\![\lambda_{d,1}\ \text{than Al is tall -er}_1\ \text{H}]\!]^r$
$= \lambda d.\ [\![\text{than Al is tall -er}_1\ \text{H}]\!]^{r[1,\textsc{the}_d]}$
$= \lambda d.\ [\textsc{d-somenon}]^{r[1,\textsc{the}_d]}_1(\top)(\lambda d'.\ \text{height}(\text{al}) \geq d')$
$= \lambda d.\ \textsc{the}_d(\top)(\lambda d'.\ \text{height}(\text{al}) \geq d')$
$= \lambda d : \top(d).\ \text{height}(\text{al}) \geq d$
$= \lambda d.\ \text{height}(\text{al}) \geq d$

The result of lambda abstraction then composes with the H with which it late merges. Since the latter denotes the vacuously true $\top$, this intersection by Predicate Modification has no effect:

(119) $[\![\text{H}\ \lambda_{d,1}\ \text{than Al is tall -er}_1\ \text{H}]\!]^r$
$= [\![\text{H}]\!]^r \cap [\![\lambda_{d,1}\ \text{than Al is tall -er}_1\ \text{H}]\!]^r$
$= \top \cap (\lambda d.\ \text{height}(\text{al}) \geq d)$
$= \lambda d.\ \text{height}(\text{al}) \geq d$

This is then fed into the higher copy of *-er*:

(120) $[\![\text{-er}_1\ \text{H}\ \lambda_{d,1}\ \text{than Al is tall -er}_1\ \text{H}]\!]^r$
$= \lambda D'.\ [\textsc{d-somenon}]^r_1(\lambda d.\ \text{height}(\text{al}) \geq d)(D')$

The derivation of the scope of *-er* proceeds identically to the derivation of its restrictor, minus the vacuous intersection with $[\![\text{H}]\!]$. As a result, we get a degree predicate true of d iff Jo is at least d-tall:

(121) $[\![\lambda_{d,1}\ \text{Jo is tall er}_1\ \text{H}]\!]^r = \lambda d.\ \text{height}(\text{jo}) \geq d$

This naturally is fed in as the scope of the degree phrase, generating the following truth conditions relative to a state cluster r:

(122) $[\![\text{Jo is taller than Al is}]\!]^r = 1$ iff
$[\textsc{d-somenon}]^r_1(\lambda d.\ \text{height}(\text{al}) \geq d)(\lambda d.\ \text{height}(\text{jo}) \geq d)$

Since we always evaluate with respect to STAY, we generate the following as the final truth conditions:

18 Given the higher types seen in Section 3, presumably $[\![\text{Al}]\!]$ should really be the $(et)t$-type $\lambda P.\ P(\text{al})$. If names can take scope via QR, then the technique outlined in Section 4.3 for $(\alpha t)t$-type quantifiers can be used to introduce state-sensitivity for names.

(123) $[\![\text{Jo is taller than Al is}]\!]^{\text{STAY}} = 1$ iff
$\textsc{d-somenon}(\lambda d.\ \text{height}(\text{al}) \geq d)(\lambda d.\ \text{height}(\text{jo}) \geq d)$
(i.e., $\exists d[\text{height}(\text{al}) \ngeq d \wedge \text{height}(\text{jo}) \geq d]$)

These are, in fact, the desired truth conditions.

To summarize, we have seen that the analysis in this paper is fully compatible with the scope-taking QR of degree phrases, even if *-er* is assigned a non-conservative interpretation. The key to our resolution of the conservativity conundrum was to follow Bhatt & Pancheva (2004) in positing that *than* clauses are late merged inside of DegP. As a result, the *than* phrase is interpreted only at the higher merge site, and not at lower merge sites, meaning it does not restrict the scope of *-er*. Thus, conservativity is not imposed, and we derive the same truth conditions as on trace-based theories of comparatives.

5 Concluding remarks

In this paper I have outlined a form of *compositional trace conversion* that generates the desired semantic effects, but without the syntactic stipulations of syntactic trace conversion or the compositional difficulties of semantic trace conversion. This analysis was shown to extend beyond the narrow purview of quantificational DPs, also being able to account for scope-taking movement by modals and degree phrases. In tying a bow on this paper, I will discuss a couple of areas that seem to me to be worth exploring in future work.

The first and perhaps most obvious issue is empirical coverage. While I have attempted to illustrate the broad applicability of the analysis in this paper by extending its scope beyond quantificational DPs, there nonetheless remain gaps that need to be filled, such as *wh-* phrases, adverbs of quantification (e.g., *always*, *usually*), and operators that quantify over focus alternatives (*only*, *even*). In addition, the analysis in this paper must be integrated with an appropriate theory of pronominal binding. These topics are left for future exploration.

Additionally, one of the primary arguments against a syntactic operation of trace conversion was that it violates the Inclusiveness Condition by inserting lexical material that does not appear in the numeration: namely, *the*. However, lambda-abstracting nodes, of which I (and others) make liberal use, also violate this condition. One path forward could be to eliminate lambda-abstracting nodes from the syntax and perform the same semantic work via a separate composition rule:

(124) Abstract and Apply (AA):
If $[\![\text{X}_n]\!]^r$ is type $(\alpha t)t$, and $[\![\text{Y}]\!]^s$ is type t, then
$[\![\text{X}_n\ \text{Y}]\!]^r = [\![\text{X}_n]\!]^r(\lambda k_\alpha.\ [\![\text{Y}]\!]^{r[n,\textsc{the}_k]})$

This rule seems to generate the right results for all of the cases at hand without the use of lambda-abstracting nodes, but questions may arise about its possible stipulativeness. While I do not find this rule particularly odious—scopal movement is common enough that it would have a wide range of applications—perhaps a more palatable alternative can be found that does not make recourse to a new rule of semantic composition. But again, this must be left for another time.

A De-parameterizing with the `Reader` applicative

In the body of the paper, state clusters were treated as a semantic parameter on a par with contexts, worlds, and times of evaluation, in spite of the fact that the former don't show the flexibility seen with these genuine parameters: we always evaluate relative to STAY. I mentioned there that a more appealing option would be for state clusters to be more directly incorporated into the compositional semantics, with a head in the left periphery contributing STAY. In this appendix I will discuss one way of accomplishing this. The approach offered here is far from the only way of getting the desired result. However, I happen to find it an elegant means of de-parameterizing swap states without overly burdening the compositional or lexical semantics.

This proposal makes use of *applicative functors* (McBride & Paterson 2008), or *applicatives* for short, which are formal devices developed in the functional programming literature. More specifically, it uses what is referred to as the `Reader` applicative. I will start by introducing applicatives in general and `Reader` in particular, and will then show how `Reader` can be used to integrate swap state clusters into the compositional semantics. It is worth noting that there are actually several ways in which `Reader` can be used to this end; I discuss the version that most appeals to my own sensibilities, but I will include footnotes where there is the potential for variation. As an illustration we will go through the semantic derivation for the LF in (5′), which is identical to (5) but with type parameters included for lambda abstractors:

(5′) $[_{\text{TP}}$ [every$_2$ teacher] $\lambda_{e,2}$ $[_{\text{TP}}$ [a$_1$ student] $\lambda_{e,1}$ T $[_{\text{VP}}$ a$_1$ student like every$_2$ teacher]]]

A.1 Applicatives and `Reader`

As mentioned in the body of the paper, the reason state clusters were treated as a parameter was because that made them easy to ignore when irrelevant and access when relevant. We would like to do more or less the same thing, while at the same time integrating state clusters into the compositional semantics. To do this we will treat state clusters as what in the functional programming literature is referred to as a *side effect*: more or less, some object or operation that occurs "on the side" during function application. Applicative functors are formal devices developed to do

precisely this: they essentially generalize function application in a way that permits the incorporation of side effects. There is a growing body of literature dedicated to demonstrating the usefulness of applicative functors in formal semantics (see, e.g., Kobele 2018, Charlow 2018, Elliott 2019); to the extent that the analysis in this appendix is appealing, it provides further support to this broader agenda.

Applicative functors consist of three main ingredients. The first is a *type constructor* $\bigcirc$, which relates an effectless type α to its effectful type $\bigcirc\alpha$. So whereas something of type *et* is a "regular" predicate, something of type $\bigcirc(et)$ is a predicate with a side effect.[19] The second ingredient is a *pure* operation $^{\uparrow}$, which takes some K of type α and returns a (vacuously) effectful $K^{\uparrow}$ of type $\bigcirc\alpha$. Finally, there is a *composition* operation $\circledast$, which takes an effectful function of type $\bigcirc(\alpha\beta)$ and an effectful input of type $\bigcirc\alpha$, and returns an effectful output of type $\bigcirc\beta$. (Here and throughout, $\circledast(K)(k)$ will be rewritten as $K \circledast k$.) Applicatives must also obey certain requirements that ensure that they are formally well-behaved; for our purposes these requirements are irrelevant, but the `Reader` applicative is known to obey them.

As mentioned previously, applicatives are a generalization of function application, meaning that "normal" function application is itself an applicative functor. More specifically, it is a rather boring applicative without side effect: $\bigcirc_{\text{fa}}\alpha$ is simply α, $K^{\uparrow_{\text{fa}}}$ is just K, and $K \circledast_{\text{fa}} k$ is simply $K(k)$. In addition to this effect-free applicative functor, we will make use of the `Reader` applicative, in which the side effect is a state argument—for our purposes, a state *cluster* argument—that is "passed upward" during function application. More specifically, the `Reader` applicative is as defined in (125), where state clusters are type r:

(125) Defining the `Reader` applicative:

a. For any type α, $\bigcirc_{\text{r}}\alpha$ is $r\alpha$.

b. For any semantic object K, $K^{\uparrow_{\text{r}}} := \lambda r.\ K$.

c. For K of type $\bigcirc_{\text{r}}(\alpha\beta)$ and k of type $\bigcirc_{\text{r}}\alpha$, $K \circledast_{\text{r}} k$ is of type $\bigcirc_{\text{r}}\beta$, and $K \circledast_{\text{r}} k := \lambda r.\ K(r)(k(r))$.

In short, for a given type α, $\bigcirc_{\text{r}}\alpha$ is the type of α-type objects with state cluster arguments "tacked on" at the beginning. The pure operator $^{\uparrow_{\text{r}}}$ tacks a vacuous state cluster argument onto a given semantic object. Finally, $\circledast_{\text{r}}$ works by applying function application while feeding both the function and the argument the same state cluster argument, thereby "passing" that state cluster argument up the tree. If `Reader` is used in our semantic derivations, we will end up with an object of type $\bigcirc_{\text{r}}t$ (i.e., rt), an effectful truth value. This can then combine with STAY via normal function application, leading to a final, state-insensitive denotation for the sentence as a whole.

19 While types are right-associative—$\alpha\beta\gamma$ is $\alpha(\beta\gamma)$, not $(\alpha\beta)\gamma$—I will treat $\bigcirc$ as left-associative: $\bigcirc\alpha\beta$ is $(\bigcirc\alpha)\beta$, not $\bigcirc(\alpha\beta)$. This is, of course, merely a matter of notational convenience.

A.2 Semantic composition with the `Reader` applicative

Some of the denotations in the body of the paper were sensitive to the state cluster parameter—namely, quantificational heads (including modals and degree heads)—and some were not, such as ⟦teacher⟧ and ⟦like⟧. With this in mind, let's suppose that those lexical items whose meanings depended on the state cluster parameter have an additional state cluster argument as their first argument:

(126) a. $[\![\text{every}_n]\!] = \lambda r \lambda P \lambda P'.\ [\textsc{every}]^r_n(P)(P')$
b. $[\![\text{a}_n]\!] = \lambda r \lambda P \lambda P'.\ [\textsc{some}]^r_n(P)(P')$

Meanwhile, lexical items whose denotations did not previously depend on the state cluster parameter do not have a state cluster argument:

(127) a. $[\![\text{teacher}]\!] = \lambda x.\ \text{teacher}(x)$
b. $[\![\text{student}]\!] = \lambda x.\ \text{student}(x)$
c. $[\![\text{like}]\!] = \lambda Q \lambda Q'.\ Q'(\lambda x.\ Q(\lambda y.\ \text{like}(x,y)))$

Our first step is to compose ⟦every$_2$⟧ with ⟦teacher⟧, but there's already a problem: the types don't match, since the former is type $r(et)(et)t$ (i.e., $\bigcirc_{\text{r}}((et)(et)t)$), while the latter is type et. Here is how we allow them to combine. First, we permit the pure operator $^{\uparrow_{\text{r}}}$—the one that introduces a vacuous state argument—to be used freely as a type shift in order to enable composition. The result of applying this to ⟦teacher⟧ is as in (128); we end up with an effectful predicate of type $\bigcirc_{\text{r}}(et)$.[20]

(128) $[\![\text{teacher}]\!]^{\uparrow_{\text{r}}}$
$= \lambda r.\ [\![\text{teacher}]\!]$
$= \lambda r \lambda x.\ \text{teacher}(x)$

Second, we include the `Reader` composition operation $\circledast_{\text{r}}$ as a means of semantic composition. This may seem undesirable, since it appears to add a new compositional rule. However, recall that "normal" function application is itself an applicative functor with composition operation $\circledast_{\text{fa}}$. What we can thus say is that there is a single rule of *generalized* function application: the grammar provides a small class of acceptable applicative composition operations, and when function application is performed the composition operation suitable to the types of the two composing denotations is used. This class can then include both $\circledast_{\text{fa}}$ and $\circledast_{\text{r}}$. Moreover, generalized function application can still be deterministic so long as the permitted composition operations have disjoint domains, as $\circledast_{\text{fa}}$ and $\circledast_{\text{r}}$ do.

20 For those who dislike a $^{\uparrow_{\text{r}}}$ type shift, an alternative could be putting an often vacuous state cluster argument in every denotation in the lexicon, so that ⟦teacher⟧, for example, would simply be $\lambda r \lambda x.\ \text{teacher}(x)$.

Since $[\![\text{every}_2]\!]$ is type $\bigcirc_{\text{r}}((et)(et)t)$, while $[\![\text{teacher}]\!]^{\uparrow_{\text{r}}}$ is type $\bigcirc_{\text{r}}(et)$, they are not in the domain of $\circledast_{\text{fa}}$, but they are in the domain of $\circledast_{\text{r}}$. Therefore, by generalized function application, we use $\circledast_{\text{r}}$ to compose the two. Importantly, note that for any semantic object K and state cluster r, $K^{\uparrow_{\text{r}}}(r) = K$; this will frequently prove useful throughout the derivation.

(129) $[\![\text{every}_2\ \text{teacher}]\!]$
$= [\![\text{every}_2]\!] \circledast_{\text{r}} [\![\text{teacher}]\!]^{\uparrow_{\text{r}}}$
$= \lambda r.\ [\![\text{every}_2]\!](r)([\![\text{teacher}]\!]^{\uparrow_{\text{r}}}(r))$
$= \lambda r.\ [\![\text{every}_2]\!](r)([\![\text{teacher}]\!])$
$= \lambda r \lambda P'.\ [\textsc{every}]^{r}_{2}(\text{teacher})(P')$

To summarize what we have so far, then, we can use the $^{\uparrow_{\text{r}}}$ operator to lift any clusterless denotation into the realm of vacuous state-sensitivity, and by generalized function application this can then compose with another state-sensitive denotation via $\circledast_{\text{r}}$ (if the types are right).[21]

This next composes with $[\![\text{like}]\!]$. Since $[\![\text{like}]\!]$ also lacks a state cluster argument, we must give it one by means of $^{\uparrow_{\text{r}}}$:

(130) $[\![\text{like}]\!]^{\uparrow_{\text{r}}}$
$= \lambda r.\ [\![\text{like}]\!]$
$= \lambda r \lambda Q \lambda Q'.\ Q'(\lambda x.\ Q(\lambda y.\ \text{like}(x,y))$

Using α as an abbreviation for the type $(et)t$, $[\![\text{like}]\!]^{\uparrow_{\text{r}}}$ is of type $\bigcirc_{\text{r}}(\alpha\alpha t)$ (i.e., $r\alpha\alpha t$), while $[\![\text{every}_2\ \text{teacher}]\!]$ is type $\bigcirc_{\text{r}}\alpha$. These are suitable types to compose via $\circledast_{\text{r}}$, so by generalized function application this is what we do:

(131) $[\![\text{like every}_2\ \text{teacher}]\!]$
$= [\![\text{like}]\!]^{\uparrow_{\text{r}}} \circledast_{\text{r}} [\![\text{every}_2\ \text{teacher}]\!]$
$= \lambda r.\ [\![\text{like}]\!]^{\uparrow_{\text{r}}}(r)([\![\text{every}_2\ \text{teacher}]\!](r))$
$= \lambda r.\ [\![\text{like}]\!]([\![\text{every}_2\ \text{teacher}]\!](r))$
$= \lambda r \lambda Q'.\ Q'(\lambda x.\ [\textsc{every}]^{r}_{2}(\text{teacher})(\lambda y.\ \text{like}(x,y)))$

Next, $[\![\text{a}_1]\!]$ and $[\![\text{student}]\!]$ compose in the same way as $[\![\text{every}_2]\!]$ and $[\![\text{teacher}]\!]$ did. Because of the parallelism, I won't show every step, and will instead simply show the result:

(132) $[\![\text{a}_1\ \text{student}]\!] = [\![\text{a}_1]\!] \circledast_{\text{r}} [\![\text{student}]\!]^{\uparrow_{\text{r}}} = \lambda r \lambda P'.\ [\textsc{some}]^{r}_{1}(\text{student})(P')$

21 For those who disprefer allowing $\circledast_{\text{r}}$ as a composition operation, the same effect can be gained by means of a type shift $^{\downarrow_{\text{r}}}$ such that for K of type $\bigcirc_{\text{r}}(\alpha\beta)$, $K^{\downarrow_{\text{r}}}$ is type $\bigcirc_{\text{r}}\alpha\bigcirc_{\text{r}}\beta$, and $K^{\downarrow_{\text{r}}} := \lambda k_{\bigcirc_{\text{r}}\alpha}.\ K \circledast_{\text{r}} k$ (cf. Kobele 2018). If $^{\downarrow_{\text{r}}}$ is applied to $[\![\text{every}_2]\!]$, the result is $\lambda\mathcal{P}_{\bigcirc_{\text{r}}(et)}.\ [\![\text{every}_2]\!] \circledast_{\text{r}} \mathcal{P}$. This can then combine with $[\![\text{teacher}]\!]^{\uparrow_{\text{r}}}$ via normal function application.

We can now compose ⟦like every$_2$ teacher⟧ with ⟦a$_1$ student⟧. Continuing with our abbreviation of α for $(et)t$, the former is type $\bigcirc_{\mathrm{r}}(\alpha t)$, while the latter is type $\bigcirc_{\mathrm{r}}\alpha$. Notice that now, both the function and the argument are already cluster-sensitive (each has an initial state cluster argument), which means that there is no need for the $^{\uparrow_{\mathrm{r}}}$ type shift and we can immediately compose via $\circledast_{\mathrm{r}}$:

(133) ⟦like every$_2$ teacher⟧ $\circledast_{\mathrm{r}}$ ⟦a$_1$ student⟧
$= \lambda r.\ ⟦\text{like every}_2\ \text{teacher}⟧(r)(⟦\text{a}_1\ \text{student}⟧(r))$
$= \lambda r.\ [\textbf{SOME}]^r_1(\text{student})(\lambda x.\ [\textbf{EVERY}]^r_2(\text{teacher})(\lambda y.\ \text{like}(x,y)))$

This gives us a denotation of type $\bigcirc_{\mathrm{r}} t$ $(= rt)$, a truth value with a state cluster argument side effect.

Next up is lambda abstraction. Because swap state clusters are now a part of the compositional semantics instead of being a semantic parameter, lambda abstraction no longer needs a separate rule of composition, and can instead be accomplished via normal function application: $⟦\lambda_{\alpha,n}⟧$ is of type $\bigcirc_{\mathrm{r}} t \bigcirc_{\mathrm{r}}(\alpha t)$, i.e., it takes an effectful truth value and returns an effectful αt-type predicate:

(134) $⟦\lambda_{\alpha,n}⟧ = \lambda T_{\bigcirc_{\mathrm{r}} t} \lambda r \lambda k_{\alpha}.\ T(r[n, \textbf{THE}_k])$

Since ⟦a$_1$ student like every$_2$ teacher⟧ is type $\bigcirc_{\mathrm{r}} t$, while $⟦\lambda_{e,1}⟧$ is type $\bigcirc_{\mathrm{r}} t \bigcirc_{\mathrm{r}}(et)$, by generalized function application we now compose via $\circledast_{\mathrm{fa}}$, rather than $\circledast_{\mathrm{r}}$. The result can be seen in (135):

(135) ⟦$\lambda_{e,1}$ a$_1$ student like every$_2$ teacher⟧
$= ⟦\lambda_{e,1}⟧ \circledast_{\mathrm{fa}} ⟦\text{a}_1\ \text{student like every}_2\ \text{teacher}⟧$
$= ⟦\lambda_{e,1}⟧(⟦\text{a}_1\ \text{student like every}_2\ \text{teacher}⟧)$
$= \lambda r \lambda z.\ ⟦\text{a}_1\ \text{student like every}_2\ \text{teacher}⟧(r[1, \textbf{THE}_z])$
$= \lambda r \lambda z.\ [\textbf{SOME}]^{r[1,\textbf{THE}_z]}_1(\text{student})$
$(\lambda x.\ [\textbf{EVERY}]^{r[1,\textbf{THE}_z]}_2(\text{teacher})(\lambda y.\ \text{like}(x,y)))$
$= \lambda r \lambda z.\ \textbf{THE}_z(\text{student})(\lambda x.\ [\textbf{EVERY}]^r_2(\text{teacher})(\lambda y.\ \text{like}(x,y)))$
$= \lambda r \lambda z : \text{student}(z).\ [\textbf{EVERY}]^r_2(\text{teacher})(\lambda y.\ \text{like}(z,y))$

We get what we were hoping for, namely, the same predicate as was generated in the body of the paper, but with the swap state cluster as an initial argument rather than a parameter of interpretation. Since ⟦a$_1$ student⟧ is type $\bigcirc_{\mathrm{r}}((et)t)$, while the result of lambda abstraction is type $\bigcirc_{\mathrm{r}}(et)$, by generalized function application composition occurs via $\circledast_{\mathrm{r}}$:

(136) ⟦a$_1$ student λ_1 a$_1$ student like every$_2$ teacher⟧
$= ⟦\text{a}_1\ \text{student}⟧ \circledast_{\mathrm{r}} ⟦\lambda_1\ \text{a}_1\ \text{student like every}_2\ \text{teacher}⟧$
$= \lambda r.\ ⟦\text{a}_1\ \text{student}⟧(r)(⟦\lambda_1\ \text{a}_1\ \text{student like every}_2\ \text{teacher}⟧(r))$
$= \lambda r.\ [\textbf{SOME}]^r_1(\text{student})(\lambda z : \text{student}(z).\ [\textbf{EVERY}]^r_2(\text{teacher})(\lambda y.\ \text{like}(z,y)))$

We next lambda abstract again, this time via $\lambda_{e,2}$:

(137) $[\![\lambda_{e,2}]\!] = \lambda T \lambda r \lambda x.\ T(r[2, \textbf{THE}_x])$

(138) $[\![\lambda_{e,2}\ \text{a}_1\ \text{student}\ \lambda_{e,1}\ \text{a}_1\ \text{student like every}_2\ \text{teacher}]\!]$
$= [\![\lambda_{e,2}]\!] \circledast_{\text{fa}} [\![\text{a}_1\ \text{student}\ \lambda_{e,1}\ \text{a}_1\ \text{student like every}_2\ \text{teacher}]\!]$
$= [\![\lambda_{e,2}]\!]([\![\text{a}_1\ \text{student}\ \lambda_{e,1}\ \text{a}_1\ \text{student like every teacher}]\!])$
$= \lambda r \lambda x.\ [\textbf{SOME}]_1^{r[2, \textbf{THE}_x]}(\text{student})$
$(\lambda z : \text{student}(z).\ [\textbf{EVERY}]_2^{r[2, \textbf{THE}_x]}(\text{teacher})(\lambda y.\ \text{like}(z,y)))$
$= \lambda r \lambda x.\ [\textbf{SOME}]_1^{r}(\text{student})(\lambda z : \text{student}(z).\ \textbf{THE}_x(\text{teacher})(\lambda y.\ \text{like}(z,y)))$
$= \lambda r \lambda x : \text{teacher}(x).\ [\textbf{SOME}]_1^{r}(\text{student})(\lambda z : \text{student}(z).\ \text{like}(z,x))$

Once again, we have an effectful predicate (type $\bigcirc_{\text{r}}(et)$) which composes with $[\![\text{every}_2$ teacher$]\!]$ by means of $\circledast_{\text{r}}$:

(139) $[\![\text{every}_2\ \text{teacher}\ \lambda_{e,2}\ \text{a}_1\ \text{student}\ \lambda_{e,1}\ \text{a}_1\ \text{student like every}_2\ \text{teacher}]\!]$
$= [\![\text{every}_2\ \text{teacher}]\!] \circledast_{\text{r}} [\![\lambda_{e,2}\ \text{a}_1\ \text{student}\ \lambda_{e,1}\ \text{a}_1\ \text{student like every}_2\ \text{teacher}]\!]$
$= \lambda r.\ [\![\text{every}_2\ \text{teacher}]\!](r)$
$([\![\lambda_{e,2}\ \text{a}_1\ \text{student}\ \lambda_{e,1}\ \text{a}_1\ \text{student like every}_2\ \text{teacher}]\!](r))$
$= \lambda r.\ [\textbf{EVERY}]_2^{r}(\text{teacher})$
$(\lambda x : \text{teacher}(x).\ [\textbf{SOME}]_1^{r}(\text{student})(\lambda z : \text{student}(z).\ \text{like}(z,x)))$

As promised, we are left with a denotation of type rt, i.e., $\bigcirc_{\text{r}} t$, an effectful truth value. Now suppose that some head in the left periphery—say, C—simply has STAY as its denotation. In this case, our denotation thus far and $[\![\text{C}]\!]$ can compose by means of normal function application ($\circledast_{\text{fa}}$), generating the final truth conditions:

(140) $[\![\text{C}]\!] = \textsc{stay}$

(141) $[\![\text{C every}_2\ \text{teacher}\ \lambda_{e,2}\ \text{a}_1\ \text{student}\ \lambda_{e,1}\ \text{a}_1\ \text{student like every}_2\ \text{teacher}]\!]$
$= [\![\text{every}_2\ \text{teacher}\ \lambda_{e,2}\ \text{a}_1\ \text{student}\ \lambda_{e,1}\ \text{a}_1\ \text{student like every}_2\ \text{teacher}]\!] \circledast_{\text{fa}} [\![\text{C}]\!]$
$= [\![\text{every}_2\ \text{teacher}\ \lambda_{e,2}\ \text{a}_1\ \text{student}\ \lambda_{e,1}\ \text{a}_1\ \text{student like every}_2\ \text{teacher}]\!]([\![\text{C}]\!])$
$= 1$ iff $\textbf{EVERY}(\text{teacher})$
$(\lambda x : \text{teacher}(x).\ \textbf{SOME}(\text{student})(\lambda z : \text{student}(z).\ \text{like}(z,x)))$
$= 1$ iff $\textbf{EVERY}(\text{teacher})(\lambda x.\ \textbf{SOME}(\text{student})(\lambda z.\ \text{like}(z,x)))$

This concludes the successful semantic derivation of *A student likes every teacher* on its inverse scope reading. We have avoided treating swap state clusters as semantic parameters and instead treated them as part of the compositional semantics. As mentioned previously, this is far from the only way of achieving this result; however, it does present what I take to be a particularly elegant solution.

References

Bhatt, Rajesh & Roumyana Pancheva. 2004. Late merger of degree clauses. *Linguistic Inquiry* 35(1). 1–45.

Bresnan, Joan. 1973. Syntax of the comparative clause construction in English. *Linguistic Inquiry* 4(3). 275–343.

Champollion, Lucas. 2015. The interaction of compositional semantics and event semantics. *Linguistics and Philosophy* 38(1). 31–66.

Charlow, Simon. 2018. A modular theory of pronouns and binding. Rutgers University, Ms.

Chierchia, Gennaro. 1995. Lecture notes from a talk given at Utrecht University.

Chomsky, Noam. 1957. *Syntactic structures*. Berlin: Walter de Gruyter.

Chomsky, Noam. 1995. *The minimalist program*. Cambridge, MA: MIT Press.

Elliott, Patrick D. 2019. Applicatives for anaphora and presupposition. Leibniz-Center for General Linguistics (ZAS), Ms.

von Fintel, Kai. 1994. *Restrictions on quantifier domains*. Amherst, MA: University of Massachusetts Amherst dissertation.

Fox, Danny. 2000. *Economy and semantic interpretation*. Cambridge, MA: MIT Press.

Fox, Danny. 2002. Antecedent-contained deletion and the copy theory of movement. *Linguistic Inquiry* 33(1). 63–96.

Fox, Danny. 2003. On logical form. In Randall Hendrick (ed.), *Minimalist syntax*, 82–123. Oxford: Blackwell Publishers.

Fox, Danny & Kyle Johnson. 2016. QR is restrictor sharing. In Kyeong-min Kim et al. (eds.), *Proceedings of the 33rd West Coast Conference on Formal Linguistics*, 1–16. Somerville, MA: Cascadilla Proceedings Project.

Fox, Danny & Jon Nissenbaum. 1999. Extraposition and scope: A case for overt QR. In Sonya Bird et al. (eds.), *Proceedings of the 18th West Coast Conference on Formal Linguistics*, 132–144. Somerville, MA: Cascadilla Proceedings Project.

Gajewski, Jon. 2002. L-analyticity and natural language. MIT, Ms.

Gajewski, Jon. 2008. More on quantifiers in comparative clauses. In Tova Friedman & Satoshi Ito (eds.), *Semantics and linguistic theory (SALT)*, vol. 18, 340–357.

Gärtner, Hans-Martin. 2002. *Generalized transformations and beyond: Reflections on minimalist syntax*. Berlin: Akademie Verlag.

Hacquard, Valentine. 2006. *Aspects of modality*. Cambridge, MA: MIT dissertation.

Heim, Irene. 1985. Notes on comparatives and related matters. University of Texas at Austin, Ms.

Heim, Irene. 2000. Degree operators and scope. In B. Jackson & T. Matthews (eds.), *Semantics and linguistic theory (SALT)*, vol. 10, 40–64.

Heim, Irene. 2006. *Little*. In Masayuki Gibson & Jonathan Howell (eds.), *Semantics and linguistic theory (SALT)*, vol. 16, 35–58.

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

Iatridou, Sabine & Hedde Zeijlstra. 2013. Negation, polarity, and deontic modals. *Linguistic Inquiry* 44(4). 529–568.

Johnson, Kyle. 2012. Towards deriving differences in how *wh* movement and QR are pronounced. *Lingua* 122(6). 529–553.

Keenan, Edward L. & Jonathan Stavi. 1986. A semantic characterization of natural language determiners. *Linguistics and Philosophy* 9(6). 253–326.

Kobele, Gregory M. 2018. The cooper storage idiom. *Journal of Logic, Language and Information* 27(2). 95–131.

Kratzer, Angelika. 1981. The notional category of modality. In H.J. Eikmeyer & H. Rieser (eds.), *Words, worlds, and contexts: New approaches in word semantics*, 38–74. Berlin: de Gruyter.

Kratzer, Angelika. 1991a. Conditionals. In Arnim von Stechow & Dieter Wunderlich (eds.), *Semantik/Semantics: An international handbook of contemporary research*, 651–656. Berlin: de Gruyter.

Kratzer, Angelika. 1991b. Modality. In Arnim von Stechow & Dieter Wunderlich (eds.), *Semantik/Semantics: An international handbook of contemporary research*, 639–650. Berlin: de Gruyter.

Kratzer, Angelika. 1998. More structural analogies between pronouns and tenses. In Devon Strolovitch & Aaron Lawson (eds.), *Semantics and linguistic theory (SALT)*, vol. 8, 92–110.

Kratzer, Angelika. 2012. *Modals and conditionals: New and revised perspectives*. Oxford: Oxford University Press.

Lassiter, Daniel. 2011. *Measurement and modality: The scalar basis of modal semantics*. New York, NY: New York University dissertation.

Lebeaux, David. 1990. Relative clauses, licensing, and the nature of the derivation. In Juli Carter et al. (eds.), *Proceedings of NELS 20*, 318–332. Amherst, MA: University of Massachusetts GLSA.

Mayr, Clemens & Benjamin Spector. 2010. Not too strong! Generalizing the Scope Economy Condition. In Martin Prinzhorn, Viola Schmitt & Sarah Zobel (eds.), *Proceedings of Sinn und Bedeutung 14*, 305–321. Vienna: University of Vienna.

McBride, Conor & Ross Paterson. 2008. Applicative programming with effects. *Journal of Functional Programming* 18(1). 1–13.

Moulton, Keir. 2015. CPs: Copies and compositionality. *Linguistic Inquiry* 46(2). 305–342.

Nunes, Jairo. 2004. *Linearization of chains and sideward movement*. Cambridge, MA: MIT Press.

Percus, Orin. 2000. Constraints on some other variables in syntax. *Natural Language Semantics* 8(3). 173–229.
Romoli, Jacopo. 2015. A structural account of conservativity. *Semantics-Syntax Interface* 2(1). 28–57.
Sauerland, Uli. 1998. *On the making and meaning of chains*. Cambridge, MA: MIT dissertation.
Sauerland, Uli. 2004. The interpretation of traces. *Natural Language Semantics* 12(1). 63–127.
Schwarzschild, Roger. 2008. The semantics of comparatives and other degree constructions. *Language and Linguistics Compass* 2(2). 308–331.
Starke, Michal. 2001. *Move dissolves into merge: a theory of locality*. Geneva: University of Geneva dissertation.
von Stechow, Arnim. 1984. Comparing semantic theories of comparison. *Journal of Semantics* 3(1). 1–77.
Williams, Edwin. 1974. *Rule ordering in syntax*. Cambridge, MA: MIT dissertation.

Robert Pasternak
Leibniz-Center for General Linguistics (ZAS)
Schützenstraße 18
10117 Berlin, Germany
pasternakrs@gmail.com