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*. In spite of their empirical and conceptual advantages, multiple-merge theories pose known challenges for the semantic interpretation of movement, 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 and 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, trace conversion, quantifiers, modals, comparatives, compositionality

Word count: 16,725 (including abstract, acknowledgments, and references)

1 Introduction

For much of the history of generative syntax, *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. For example, consider (1) on its inverse scope (*every* > *a*) interpretation:

* 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 audiences at the ZAS Semantics and Pragmatics Reading Group, the University of Göttingen's Oberseminar English Linguistics, and the LSA 2020 Annual Meeting. Comments from Kjell Johan Sæbø and three anonymous reviewers were extremely valuable. Special thanks to Patrick Elliott for many long and fruitful conversations in the early stages of this project. All remaining errors are mine. This research is funded by DFG Grant #387623969 (*DP-Border*, PIs: Artemis Alexiadou and Uli Sauerland).

(1) A student likes every teacher.

Here *every teacher* seems to simultaneously occupy both its overt position as the internal argument of *like*, and a higher position at which it outscopes *a student*. This is typically explained through *quantifier raising (QR)*: *every teacher* covertly moves past *a student*, leaving a trace that is interpreted as a bound variable argument to *like*:

(2) $[_{TP}$ every$_2$ teacher λ_2 $[_{TP}$ a$_1$ student λ_1 T $[_{VP}$ t_1 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 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 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", X is in fact 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*.

Under the multiple-merge umbrella, two main candidate theories have emerged. 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 (2) would be replaced with (3):

(3) $[_{TP}$ 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 below Y

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.

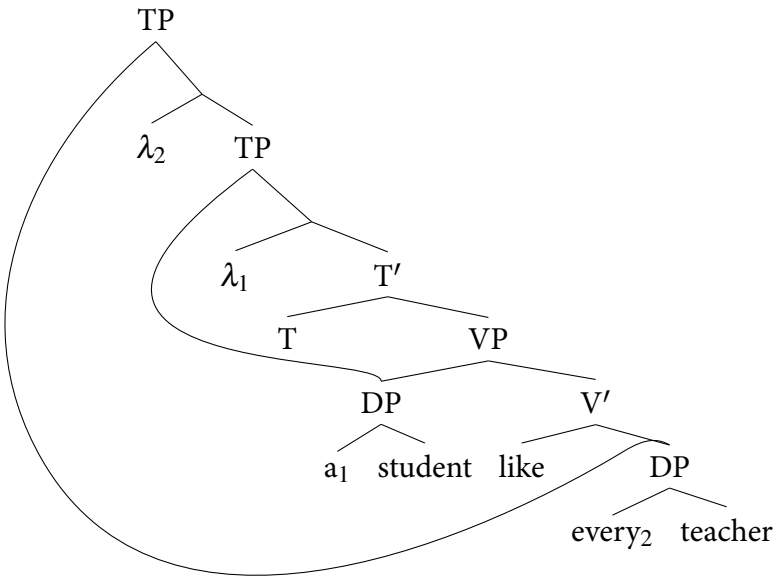

Figure 1 Multidominance LF for inverse scope of *A student likes every teacher*.

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, it is quite literally the same constituent that sits in both positions. Thus, the translation of the LF in (2) into the multidominance theory of movement will look like Fig. 1.

In spite of their empirical and conceptual advantages, multiple-merge theories pose well-known challenges for traditional approaches to semantic composition. A common means of interpreting LFs like (2) 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 (3) and Fig. 1, as it would require that a DP be interpreted as a true quantifier at its highest merge site, and as a bound variable at lower merge sites. 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 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:[2]

(4) (3) AFTER TRACE CONVERSION:

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

We can then posit that the denotation of *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, $[\![\text{the}_n]\!](P)$ denotes a *restricted variable*:

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

For example, $[\![\text{the}_1\ \text{student}]\!]^g$ 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. Hence, replacing *a*$_1$ *student* and *every*$_2$ *teacher* with *the*$_1$ *student* and *the*$_2$ *teacher* permits precisely the sort of bound variable reading required for successful composition. 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 discuss a specific semantic prediction of trace conversion: when a DP undergoes movement, its NP restrictor semantically contributes not only at the DP's highest merge site, but also at its lower merge sites in the form of a domain restriction. I will also briefly discuss some empirical arguments that have been made in favor of this hypothesis, and thus in favor of multiple-merge over trace-based theories of movement. However, in Section 3 I go over some critical downsides to trace conversion, based on a mixture of empirical observations, theory-internal considerations, and broader principles of theoretical simplicity and elegance. In brief, syntactic trace conversion is difficult to motivate on independent syntactic grounds and requires abandoning well-motivated syntactic principles (namely, Chomsky's (1995) *Inclusiveness Condition*), as well as additional stipulations to account for non-DP scope-taking. Meanwhile, semantic trace conversion runs afoul of basic principles of semantic compositionality, since the same DP must be interpreted differently in different locations. The goal in the rest of the paper will be to retain the benefits of trace conversion without these downsides.

In Section 4 I turn to my own analysis, which I refer to as *compositional trace conversion*, and which allows LFs like (3) and Fig. 1 to be interpreted compositionally and without syntactic modifications. Put simply, the semantic impact of trace

2 The particular implementation adopted here is that of Sauerland (2004). Differences between this and Fox's version are important, but not for the purposes of the argumentation in this paper.

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 5, I show how the 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 6.[3]

2 The Interpreted Lower Restrictor Hypothesis

Before discussing the downsides to trace conversion, it is worth going over one of its positives. Consider again the post-trace-conversion LF in (4). As discussed previously, *the*$_1$ *student* denotes a restricted variable: $g(1)$ iff $g(1)$ is a student, and otherwise undefined. Likewise for *the*$_2$ *teacher*:

(6) $[\![\text{the}_1 \text{ student like the}_2 \text{ teacher}]\!]^g$ is defined iff $\texttt{student}(g(1))$ and $\texttt{teacher}(g(2))$.
Where defined, $[\![\text{the}_1 \text{ student like the}_2 \text{ teacher}]\!]^g = \texttt{like}(g(1), g(2))$

Because *the*$_1$ *student* denotes a restricted variable, when we lambda-abstract over that variable we get a predicate whose domain is restricted to students:

(7) $[\![\lambda_1 \text{ the}_1 \text{ student like the}_2 \text{ teacher}]\!]^g$ is defined iff $\texttt{teacher}(g(2))$.
Where defined, $[\![\lambda_1 \text{ the}_1 \text{ student like the}_2 \text{ teacher}]\!]^g =$
$\lambda x : \texttt{student}(x).\ \texttt{like}(x, g(2))$

In other words, because of the definition of $[\![\text{the}_n]\!]^g$, we predict NP restrictors in the "traces" of DP movement to make semantic contributions in the form of domain restrictions. I will call this the **Interpreted Lower Restrictor Hypothesis** (**ILRH**):

(8) Interpreted Lower Restrictor Hypothesis (ILRH):
When a DP of the form $[_{\text{DP}}$ D NP$]$ undergoes movement, NP is also semantically interpreted at the trace position, so that after lambda abstraction the resulting predicate is restricted to individuals in $[\![\text{NP}]\!]$.

Several empirical arguments have been offered in favor of ILRH. For example, Erlewine (2014) provides evidence from association with focus—in short, it seems that focus-sensitive operators can associate with focused material inside of traces—and Romoli (2015) follows Chierchia (1995), Fox (2002), and Sportiche (2005) in using

3 For reasons of space I cannot discuss other attempts at a Minimalism-friendly semantics, which make more fundamental revisions to traditional assumptions about the syntax-semantics of quantification. These include Gotham's (2018) LF-less "Glue semantic" approach, as well as the analyses of Johnson (2012) and Fox & Johnson (2016) in which QR does not involve movement of quantificational heads. Comparing these theories is of course important work; my analysis can be thought of as an attempt to bring the traditional approach closer to its best form in order to facilitate such comparisons.

ILRH to account for the famed *conservativity hypothesis* (Barwise & Cooper 1981, Keenan & Stavi 1986). However, to keep things brief I will only discuss ILRH in relation to the treatment by Sauerland (1998, 2004) of certain puzzling facts pertaining to antecedent-contained deletion (ACD), augmented by Fox's (2002) proposal connecting ACD to extraposition. This choice is motivated by two factors: the empirical puzzle provides particularly compelling evidence for ILRH, and going over the analysis will help illustrate some key concepts that will prove useful in later discussion.

2.1 Setting the table: ACD and the Kennedy-Sauerland Puzzle

Antecedent-contained deletion refers to ellipsis sites that, at least by appearances, are contained within their own antecedents. Consider (9):

(9) Lisa read every book that Anna did.

The elided VP is inside the relative clause, and its antecedent is the matrix VP. The apparent containment of the ellipsis within its own antecedent is illustrated in (10):

(10) Lisa $\overbrace{\text{read every book that Anna did } \underbrace{\text{(read)}}_{\text{elided VP}}}^{\text{antecedent VP}}$

If the ellipsis is truly contained within its own antecedent, this poses an apparent problem: the ellipsis and antecedent VP cannot match without an infinite regress. To avoid this, traditional accounts of ACD posit that the DP headed by *every* QRs outside of the VP (Sag 1976, May 1985):

(11) [every$_1$ book that Anna did Δ] λ_1 Lisa read t_1

With the ellipsis site no longer contained within its antecedent, ellipsis resolution can be achieved without any infinite regress.

The empirical evidence for ILRH that we will be discussing comes from a particular puzzle concerning ACD, which I will call the *Kennedy-Sauerland Puzzle*:

(12) Kennedy-Sauerland Puzzle (Sauerland 2004: p. 64):
  a. * Polly visited every town that is near the lake Erik did.
  b. Polly visited every town that is near the one Erik did.

The first half of the puzzle, initially observed by Kennedy (1994), is the ill-formedness of (12a). This sentence is well-formed without the VP ellipsis, indicating that its ill-formedness is due specifically to a lack of ellipsis licensing.

(13) Polly visited every town that is near the lake Erik visited.

But on traditional analyses there is no reason to suspect that (12a) should be ill-formed: in the LF in (14), the ellipsis is outside of its antecedent, just as it is in (11).

(14) [every$_1$ town that is near the lake Erik did Δ] λ_1 Polly visited t_1

The second half of the puzzle, noted by Sauerland (1998, 2004), is the fact that (12b)—identical to (12a) except that *lake* is replaced by *one*—is well-formed. This indicates that whether or not ellipsis is licensed in examples like (12) must be sensitive to the choice of noun that the most deeply embedded relative clause adjoins to (*lake* vs. *one*), an even stranger result on traditional analyses.

Sauerland (1998, 2004) offers an account of this puzzle that hinges on ILRH. But before going into the specifics of Sauerland's solution, it will help to first go over a particular proposal by Fox (2002) connecting ACD to extraposition.

2.2 Fox 2002 and the ACD-extraposition connection

Fox (2002) follows Baltin (1987) in arguing that ACD necessarily involves (often string-vacuous) *extraposition*, the same process separating the relative clause from *book* in (15):[4,5]

(15) I read [every book] yesterday [that John had recommended].

He adopts Fox & Nissenbaum's (1999) analysis of extraposition, which crucially relies on multiple-merge. An important empirical observation about extraposition is what Fox refers to as *Williams's Generalization* (after Williams (1974)), which states that any scope-taker must scope at least as high as any adjunct extraposed from it. An illustration of Williams's Generalization can be seen in (16):

(16) 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.

The extraposition-less (16a) is ambiguous. If *every* scopes below *before*, the resulting interpretation is that I was the first to make my way through John's list, though certain books you might have finished first. If *every* outscopes *before*, the interpretation is stronger: each book was finished by me first. But (16b), in which the relative clause is extraposed past *before*, is unambiguous, and only the latter reading is available.

In their analysis of extraposition, Fox & Nissenbaum (1999) follow Lebeaux (1990) in positing that adjuncts can be *late merged*, i.e., adjoined to the constituent they modify after that constituent has already merged into a larger structure and undergone movement. Fox & Nissenbaum further propose that late merger can occur after

4 For arguments against Baltin's analysis, see Larson & May 1990. See Fox 2002 for counterarguments.
5 Fox & Nissenbaum (1999) note that both nominal arguments and adjuncts can undergo extraposition. However, argument and adjunct extraposition seem to be two distinct processes; the empirical findings and analysis under discussion are specific to adjunct extraposition.

QR, and that this is what happens in the case of extraposition. In (16b), for example, *every book* QRs past the *before*-phrase, with the relative clause adjoining to the higher, unpronounced copy of *book*. If QR is stipulated to be rightward, this generates the correct word order:

(17) [I read [every$_1$ book] before you did] [every$_1$ book that John…]

Importantly, by tying extraposition to QR in this fashion, Williams's Generalization can be accounted for. After all, in order to generate the extraposed structure, *every book* had to QR past the *before*-phrase. Trace conversion then converts this structure into one that is semantically interpretable, and the interpretation is one in which *every* outscopes *before*:

(18) [λ_1 I read the$_1$ book before you did] [every$_1$ book that John …]

Therefore, Fox & Nissenbaum (1999) rightly predict that the only way to generate the string in (16b) is by means of a structure in which *every book* outscopes the *before*-phrase, thereby eliminating the ambiguity seen in (16a).

Fox (2002) proposes that this analysis of extraposition be extended to ACD examples like (9): the ellipsis-containing relative clause is late merged after QR, leading to (string-vacuous) extraposition and the approximate LF structure seen in (19), with gaps to be filled in shortly:

(19) [λ_1 Lisa read the$_1$ book] [every$_1$ book that Anna did]

Thus, on this analysis "antecedent-contained deletion" is in fact a misnomer: since the relative clause containing the ellipsis is late merged after QR, there is actually no point at which the ellipsis is contained within its antecedent.

Many of Fox's arguments for connecting ACD to extraposition in this manner are beyond the scope of this paper. However, one such argument comes precisely from its utility when combined with Sauerland's (1998, 2004) account of the Kennedy-Sauerland puzzle, to which we now turn.

2.3 Resolving the Kennedy-Sauerland Puzzle

Sauerland's solution to the Kennedy-Sauerland puzzle needs two additional ingredients. First, we adopt a *PF deletion* theory of ellipsis, in contrast to an *LF copying* theory: an ellipsis site is not an empty constituent that is subsequently "filled in" at LF, but a fully syntactically realized constituent that can be erased at PF under certain conditions. As for what these conditions are, Sauerland follows Rooth (1992) and Fox (1999) in connecting ellipsis licensing to contrastive focus. However, for our purposes we can adopt a simpler picture: antecedent X can license the ellipsis of Y only if X and Y are semantically identical, *modulo* indexation. Thus, *see herself*$_1$ and *see herself*$_2$ count as sufficiently similar, but not *see her*$_1$ *friend* and *see her*$_1$ *enemy*.

Second, Sauerland proposes a *matching analysis* of relative clauses: in an NP with adjoined relative clause like *lake that Erik visited*, the head noun *lake* also occurs inside the relative clause. More specifically, the internal argument of *visit* is saturated by the DP *Op lake*, where *Op* is a *wh*-like determiner analogous to *which*. This DP then undergoes movement to the left periphery of the relative clause, whereupon its noun is deleted under identity with the adjoined-to *lake*:

(20) $[_{NP}$ lake $[_{CP}$ $[_{DP}$ OP_1 ~~lake~~] λ_1 that Erik visited $[_{DP}$ OP_1 lake]]]

After trace conversion this NP will look as in (21) at LF. For convenience I ignore the higher merge site of *Op lake*, since how it is or is not integrated into the compositional semantics is not relevant for the present discussion. (However, see Section 5.3 for some discussion of the relevance of this issue to the theory proposed in this paper.)

(21) $[_{NP}$ lake $[_{CP}$ λ_1 that Erik visited the$_1$ lake]]

Thus, by combining multiple-merge, ILRH, and the matching analysis of relative clauses we predict that the semantic interpretation of the head noun should also be visible inside the relative clause, in the form of a domain restriction.

This can then be combined with Fox's (2002) extraposition-based analysis of ACD, leading to the post-trace-conversion LF for (9) shown in (22):

(22) $[\lambda_1$ Lisa read [the$_1$ book]]
[every$_1$ book [(OP_2 ~~book~~) λ_2 that Anna read the$_2$ book]]

Notice that the antecedent (*read the*$_1$ *book*) and the ellipsis site (*read the*$_2$ *book*) are indeed semantically identical *modulo* indexation. Hence, we rightly predict the availability of ellipsis and the grammaticality of (9).

By adopting these assumptions, the Kennedy-Sauerland facts in (12) fall out immediately. Let us start with the ill-formed (12a), which we predict to have the following post-trace-conversion LF:

(23) $[\lambda_1$ Polly $\overbrace{\text{visited the}_1 \text{ town}}^{\text{antecedent}}$]
[every$_1$ town (OP_2 ~~town~~) λ_2 that the$_2$ town is near
[the lake (OP_3 ~~lake~~) λ_3 Erik $\underbrace{\text{visited the}_3 \text{ lake}}_{\text{*elided}}$]]

This is a perfectly well-formed LF structure, hence (13). However, ellipsis is not licensed. The antecedent *visited the*$_1$ *town* is not semantically identical to the elided *visited the*$_3$ *lake*, regardless of indexation: the former has an argument saturated by a variable restricted to towns, while the latter has an argument saturated by a variable restricted to lakes. So (12a) is ill-formed.

(12b), meanwhile, will have the same LF structure, but replacing *lake* with *one*. Regardless of whether one thinks of *one* as an NP pronoun anaphoric to *town* or

as an instance of *town* that is converted to *one* at PF, it is clear that in this example ⟦one⟧ = ⟦town⟧. Hence, the antecedent *visited the*$_1$ *town* and the elided *visited the*$_3$ *one* are semantically identical *modulo* indexation—in both, the internal argument is saturated by a variable restricted to towns—and ellipsis is licensed in (12b).

Notice that ILRH is critical to this account of the Kennedy-Sauerland Puzzle. After all, both the antecedent and elided (or unelidable) constituents contain traces (of *every town* and *Op lake/one*, respectively), and the only way to semantically differentiate between these traces—thereby preventing (12a) and permitting (12b)—is to posit that *town*/*lake*/*one* makes a semantic contribution at its lower merge site. Since trace conversion adheres to ILRH, it makes for a useful framework in which to formulate Sauerland's analysis. Therefore, given the arguments against trace conversion that will be offered in the next section, it will be important that whatever ends up replacing it similarly adhere to ILRH.

3 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.

3.1 Syntactic trace conversion

The most popular approach to trace conversion, and the version assumed thus far, is *syntactic*: a post-syntactic operation replaces determiners at lower merge sites with *the*$_n$. Note that an additional benefit to trace conversion beyond its semantic results is that it swaps in a syntactic object whose existence has already been motivated on 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 that adheres to ILRH, 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.[6] In other words, trace conversion is an inherently countercyclic operation that replaces certain constituents with semantically interpreted material absent from the numeration, a significant violation of Chomsky's (1995) otherwise robust Inclusiveness Condition. While Inclusiveness is an empirical hypothesis and should not be taken as gospel, on basic Minimalist principles the prospect of abandoning it should at least give us pause. After all, if we wish to determine the extent to which the language faculty is optimally designed—including how much of its behavior can be predicted by attributing to it only basic operations like Merge and Agree—then introducing further operations like syntactic trace conversion that make it suboptimal in this regard should not be done unless necessary.

Even if we put Inclusiveness aside, trace conversion remains a rather strange syntactic operation, and it is not entirely clear how its existence could be empirically motivated. It will help to compare and contrast it with another covert syntactic operation: covert movement. The fact that the standard approach to the semantics of quantification happens to require movement to effect scope reconfiguration does not, of course, constitute evidence for covert movement: this is an observation about a theory, not about language. This is why researchers have looked elsewhere for evidence of covert movement, including parallels between covert and overt movement, cross-linguistic variation in whether certain movement operations are covert or overt, effects of covert movement on overt structure (e.g., Fox & Nissenbaum's (1999) aforementioned account of extraposition), etc. Yet to my knowledge the only existing motivations for syntactic trace conversion involve its necessity under current semantic assumptions for generating appropriate interpretations (including ILRH). But again, this is an observation about a semantic theory, not an observation about language. Moreover, the kinds of evidence in favor of covert movement do not seem to be available for trace conversion: I am unaware of any evidence for the existence of an overt counterpart to trace conversion, nor of trace conversion affecting overt structure. Put succinctly, if contemporary semantic assumptions happened to mesh well with multiple-merge and generate the appropriate interpretations, there would be no reason to posit a syntactic operation of trace conversion at all.

This implementation 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 (24) and (25) 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*:

6 *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.

(24) (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*)

(25) (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 "trace") in their merge positions.

(26) 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.[7] 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.

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)*:

(27) Category-Neutral Trace Conversion (Moulton 2015: 326):
   a. Quantifier Removal: [$_{DP}$ every square]$_3$ ⇝ [$_{DP}$ square]$_3$
   b. Index Interpretation: [$_{DP}$ square]$_3$ ⇝ [$_{DP}$ 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):

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

7 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 (24) and (25) it is *-er* that must be replaced with *the*.

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), as a syntactic operation Index Interpretation is undesirable for the same reasons as the original formulation of trace conversion: it violates Inclusiveness and is motivated only on the grounds that it is necessary to maintain current semantic assumptions. 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:

(29) $[\![[_{\text{XP}}\ n : n \text{ is (a) Y}]]\!]^g = g(n)$ 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:

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

This would give us a version of trace conversion that is category-neutral, but that does not require a new compositional rule. But in addition to retaining the syntactic disadvantages of the original formulations of trace conversion and Index Interpretation, 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, while syntactic trace conversion has some semantic benefits, there is reason to think it is undesirable as a syntactic operation: it violates Inclusiveness and is motivated only by its necessity under certain contingent semantic assumptions. 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 (*schme*) that has no overt counterpart, and that closely resembles the very thing multiple-merge theories of movement seek to replace: namely, traces.

3.2 Semantic trace conversion

An alternative possibility mentioned by Fox (2003) is that trace conversion 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. (See Ruys 2015 for a similar proposal.)

(31) 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.

Note that while this particular implementation does not adhere to ILRH, one could presumably formulate an equally plausible alternative that does.

Given that on a semantic trace conversion account an LF like (3) or Fig. 1 can be directly interpreted without intervening syntactic operations, this approach naturally avoids the pitfalls of its syntactic counterpart. However, in the end these problems are not so much eliminated as they are shifted from syntax to semantics, as we face a new problem of non-compositionality: the interpretation of a scope-bearing constituent is no longer a function of its structure, since the same phrase is interpreted like a trace in some syntactic environments and like a quantifier in others.

As a reviewer points out, this is not necessarily anti-compositional if we define compositionality in a way that makes use not only of the LF representation, but also of 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:

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

(33) 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:

(34) $[\![\text{faux}_n]\!]^g$ 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, $[\![\text{faux}_n]\!]^g = g(n)$.

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

Though such 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.[8]

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*.

4 Compositional trace conversion and DP movement

Our task now is to define a semantics that generates the results of trace conversion, but does so compositionally and without any syntactic modifications. We first lay out the proposal for DP quantification. Notice that as far as the compositional semantics is concerned, the copy LF in (3) and the multidominance LF in Fig. 1 are identical: there is no difference between (I) distinct but indistinguishable copies at separate merge sites, and (II) the same constituent merged at two separate locations. Any analysis that can interpret one LF directly can interpret either LF directly, including the analysis proposed in this paper. This in turn means that unlike syntactic

8 Another possibility, suggested by Ruys (2015), is that higher and lower copies can be differentiated featurally: a lower copy might have an unchecked feature that is checked in a higher copy (e.g., Case). However, this does not seem to militate against lexical items like *nonce* or *faux*, whose movements would presumably also be feature-driven. Moreover, this approach seems to conflict with basic Minimalist assumptions about the architecture of the grammar. If the feature that differentiates between higher and lower copies is semantically interpretable, it should not be deleted upon checking and should thus be present in all copies. If it is uninterpretable, then it must be deleted in all copies in order to prevent a crash: hence, uninterpretable.

trace conversion, which requires a distinction between higher copies that do not undergo trace conversion and lower copies that do, the present analysis is fully agnostic between copy theory and multidominance theory. However, for the sake of concreteness and simplicity I will often adopt the language of copy theory. In going over how the system works, we will use the LF in (3), repeated below, as our example:[9]

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

4.1 Swap states as a vehicle for trace conversion

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 *the*$_n$-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 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.[10] For a given state s, index n, and etetts E and E', if $s(n)(E) = E'$, I will say that *s swaps out E for E' at (index) n*, or equivalently, *s swaps in E' for E at (index) n*. For readability's sake, I will rewrite $s(n)(E)$ as $[E]^s_n$.

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 interpretation, 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 (36):

(36) 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, 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 gaps to be filled in later, can be seen in (37):

9 It is worth noting that on the syntactic end of things, I operate under some relatively common (though often tacit) assumptions about indices and lambda-abstractors. At the very least, I assume that when movement occurs, lambda-abstractors are inserted that are co-indexed with the moving operators. I also assume that distinct operators are assigned distinct indices, though assigning multiple operators the same index could be an intriguing way to analyze across-the-board movement (cf. Fox & Johnson 2016). Whether lambda-abstractors can be inserted without being triggered by movement is an issue left for future work. For discussion of issues related to indices and lambda-abstractors, see Section 6.

10 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 $\langle\langle\alpha,\beta\rangle,\gamma\rangle$.

(37) 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 (38):

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

I will often rewrite $\lambda x.\ \texttt{teacher}(x)$ as `teacher` when convenient, and likewise for $\lambda x.\ \texttt{student}(x)$. 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 (39):[11]

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

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:

(40) a. $[\![\text{a}_n]\!]^s = [\texttt{SOME}]^s_n = \lambda P \lambda P'.\ [\texttt{SOME}]^s_n(P)(P')$
b. $[\![\text{every}_n]\!]^s = [\texttt{EVERY}]^s_n = \lambda P \lambda P'.\ [\texttt{EVERY}]^s_n(P)(P')$

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

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

11 As a reviewer notes, $[\![\text{like}]\!]$ can also be defined as follows, while still allowing composition with two $(et)t$-type arguments:

(i) $[\![\text{like}]\!]^s = \lambda Q \lambda x.\ Q(\lambda y.\ \texttt{like}(x,y))$

To my knowledge there is no meaningful difference between this definition and (39).

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

(42) a. $[\![\text{a}_1]\!]^s([\![\text{student}]\!]^s) = \lambda P'.\ [\texttt{SOME}]^s_1(\texttt{student})(P')$

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

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. Whatever this bound variable etett is, it must be parameterized to an individual: namely, the entity argument z that is lambda-abstracted over. In keeping with standard implementations of trace conversion, I will use $\texttt{THE}_z$, defined below:

(43) $\texttt{THE}_z := \lambda P \lambda P' : P(z).\ P'(z)$

Now that we have a 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 (36), 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 $s[1,\texttt{THE}_z]$, which is the state identical to s except that $s[1,\texttt{THE}_z]$ swaps out all etetts for $\texttt{THE}_z$ at index 1. More generally:

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

With this in place, we now have the formal tools necessary in order to define lambda abstraction and fill in the blanks in (37). This can be seen in (45):

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

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

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

By definition, for any state s and etett E, $[E]^{s[1,\texttt{THE}_z]}_1 = \texttt{THE}_z$, meaning that $[\texttt{SOME}]^{s[1,\texttt{THE}_z]}_1$ can be replaced with $\texttt{THE}_z$. Similarly, for any state s, etett E, and $n \neq 1$, $[E]^{s[1,\texttt{THE}_z]}_n = [E]^s_n$, meaning that $[\texttt{EVERY}]^{s[1,\texttt{THE}_z]}_2$ can be replaced with $[\texttt{EVERY}]^s_2$.

(47) $[\![\lambda_1\ \mathrm{a}_1\ \text{student like every}_2\ \text{teacher}]\!]^s$
$= \lambda z.\ \mathtt{THE}_z(\mathtt{student})(\lambda x.\ [\mathtt{EVERY}]^s_2(\mathtt{teacher})(\lambda y.\ \mathtt{like}(x,y)))$
$= \lambda z : \mathtt{student}(z).\ [\mathtt{EVERY}]^s_2(\mathtt{teacher})(\lambda y.\ \mathtt{like}(z,y))$

Notice that crucially, this analysis adheres to ILRH, an aforementioned desideratum: after lambda abstraction, the lower instance of *student* makes a semantic contribution in the form of a domain restriction.

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

(48) $[\![\mathrm{a}_1\ \text{student}]\!]^s([\![\lambda_1\ \mathrm{a}_1\ \text{student like every}_2\ \text{teacher}]\!]^s)$
$= [\mathtt{SOME}]^s_1(\mathtt{student})(\lambda z : \mathtt{student}(z).\ [\mathtt{EVERY}]^s_2(\mathtt{teacher})(\lambda y.\ \mathtt{like}(z,y)))$

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

(49) $[\![\lambda_2\ \mathrm{a}_1\ \text{student}\ \lambda_1\ \mathrm{a}_1\ \text{student like every}_2\ \text{teacher}]\!]^s$
$= \lambda x.\ [\![\mathrm{a}_1\ \text{student}\ \lambda_1\ \mathrm{a}_1\ \text{student like every}_2\ \text{teacher}]\!]^{s[2,\mathtt{THE}_x]}$
$= \lambda x.\ [\mathtt{SOME}]^{s[2,\mathtt{THE}_x]}_1(\mathtt{student})$
$\quad(\lambda z : \mathtt{student}(z).\ [\mathtt{EVERY}]^{s[2,\mathtt{THE}_x]}_2(\mathtt{teacher})(\lambda y.\ \mathtt{like}(z,y)))$
$= \lambda x.\ [\mathtt{SOME}]^s_1(\mathtt{student})(\lambda z : \mathtt{student}(z).\ \mathtt{THE}_x(\mathtt{teacher})(\lambda y.\ \mathtt{like}(z,y)))$
$= \lambda x : \mathtt{teacher}(x).\ [\mathtt{SOME}]^s_1(\mathtt{student})(\lambda z : \mathtt{student}(z).\ \mathtt{like}(z,x))$

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

(50) $[\![\text{every}_2\ \text{teacher}]\!]^s([\![\lambda_2\ \mathrm{a}_1\ \text{student}\ \lambda_1\ \mathrm{a}_1\ \text{student like every}_2\ \text{teacher}]\!]^s)$
$= [\mathtt{EVERY}]^s_2(\mathtt{teacher})$
$\quad(\lambda x : \mathtt{teacher}(x).\ [\mathtt{SOME}]^s_1(\mathtt{student})(\lambda z : \mathtt{student}(z).\ \mathtt{like}(z,x)))$

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 (50) uses $[\mathtt{SOME}]^s_1$ and $[\mathtt{EVERY}]^s_2$ instead of just SOME and EVERY, the truth conditions of (50) 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 (51), which swaps out every etett for itself at every index:

(51) $\mathtt{Stay} := \lambda n \lambda E.\ E$

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 (52):

(52) $[\![(3)]\!]^{\texttt{Stay}} = \texttt{EVERY}(\texttt{teacher})$
$(\lambda x : \texttt{teacher}(x).\ \texttt{SOME}(\texttt{student})(\lambda z : \texttt{student}(z).\ \texttt{like}(z,x)))$

We thus have 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: instead, the semantic work of trace conversion is automatically performed by lambda abstraction. Before turning to the task of generalizing this analysis beyond DPs, I will first briefly discuss some other features of the analysis: how it can be used to define semantic reconstruction, and whether and how it might be replaced with an alternative with lower-type swap states.

4.2 A possible extension: Semantic reconstruction

Reconstruction is the process whereby a constituent takes scope at a position below its highest landing site. This is exemplified by the weaker *not* > *every* reading of (53), according to which at least one student did not leave:

(53) Every student didn't leave.

A common analysis of reconstruction is as a syntactic phenomenon. Following May (1977, 1985), traditional treatments use an operation of *quantifier lowering*, in which *every student* moves downward from its pronounced position to somewhere below negation—presumably, its VP- or *v*P-internal merge position. While syntactic lowering operations are typically no longer used, the results of quantifier lowering can still be obtained in multiple-merge theories. Thus, in a copy theory of movement, any copies of *every student* that outscope negation can be erased or ignored, furnishing an LF structure that looks the same as if *every student* had never moved past *not* in the first place. For example, while (53) would have an LF like (54a) on its surface scope interpretation, its LF would look like (54b) on its inverse scope interpretation:

(54) a. every$_1$ student λ_1 not every$_1$ student leave
b. ~~every$_1$ student~~ not every$_1$ student leave

Similarly, in a multidominance theory all connections can be severed between *every student* and points in the structure higher than negation, again generating an LF structure that looks like one in which *every student* never moves past negation. Such structures would compose in an entirely straightforward fashion on the analysis in this paper, and the current theory doesn't have much of interest to add.

However, more recently it has been argued that some instances of reconstruction—perhaps even all—are not realized syntactically, but only semantically: the movement of one operator past another is not "undone" in the syntax, but compositional mechanisms ensure that when the moved operator composes in its higher position, the semantic result is still one in which it takes low scope (von Stechow 1991, Cresti

1995, Rullmann 1995, Lechner 1998, Sharvit 1999, Wurmbrand 2010, Ruys 2015). I will not in this paper provide empirical arguments for or against the existence of semantic reconstruction, either in addition to or instead of syntactic reconstruction. However, it is worth noting that *if* semantic reconstruction exists, it can be captured fairly easily on the analysis proposed in this paper. To illustrate, I will show how an inverse scope reading of (53) can be derived by means of semantic reconstruction within the confines of the present analysis.

Traditional treatments of semantic reconstruction involve traces being interpreted as higher-type variables. Suppose that the LF for (53) looks as in (55):

(55) every$_1$ student λ_1 not t_1 leave

If the trace is interpreted as an e-type variable, the result after lambda-abstraction will be (56a); when ⟦every student⟧ takes this as an argument, the result will be a surface scope (*every* > *not*) interpretation. But if the trace is interpreted as an $(et)t$-type variable, then after lambda-abstraction the result will be the higher-type (56b). This will then take ⟦every student⟧ as its argument, generating a semantically reconstructed inverse scope (*not* > *every*) interpretation.

(56) a. $\lambda x.\ \neg\texttt{leave}(x)$
b. $\lambda Q.\ \neg Q(\texttt{leave})$

The same result can be derived on the analysis in this paper. Suppose that on top of the "normal" lambda-abstractor λ_n, there is a higher-type lambda-abstractor λ'_n, which performs semantically reconstructing lambda abstraction. (Alternatively, we could suppose that there is one lambda-abstractor λ_n, with optionality as to whether reconstructing or non-reconstructing lambda abstraction takes place.) Thus, the LF for (53) on its inverse scope interpretation will be as in (57):

(57) every$_1$ student λ'_1 not every$_1$ student leave

Up to and excluding lambda abstraction, the interpretation would be exactly as we would predict based on the analysis developed thus far:

(58) $[\![\text{not every}_1\text{ student leave}]\!]^s = \neg[\texttt{EVERY}]^s_1(\texttt{student})(\texttt{leave})$

Next we define reconstructing lambda abstraction. First, a helpful abbreviation:

(59) For J of type $\alpha\beta$, $\underline{J}$ is type $\alpha\alpha\beta$, and $\underline{J} := \lambda K_\alpha.\ J$.

$\underline{J}$ is simply J with an extra, vacuous argument tacked on that is of the same type as J's first argument. Thus, if Q is an $(et)t$-type quantifier, $\underline{Q}$ will be an etett. Reconstructing lambda abstraction can then be defined as in (60):

(60) Reconstructing Lambda Abstraction:
$[\![\lambda'_n\ \text{X}]\!]^s = \lambda Q.\ [\![\text{X}]\!]^{s[n,\underline{Q}]}$

Much like on traditional approaches to semantic reconstruction, the result of lambda abstraction in (60) takes an $(et)t$-type argument instead of an e-type argument. As discussed above, since the Q argument is type $(et)t$, $\underline{Q}$ is an etett, meaning that it can be the output to a swap state.

To see how this works, let's apply it to (58):

(61) $[\![\lambda_1'$ not every$_1$ student leave$]\!]^s$
$= \lambda Q.\ [\![$not every$_1$ student leave$]\!]^{s[1,\underline{Q}]}$
$= \lambda Q.\ \neg[\texttt{EVERY}]_1^{s[1,\underline{Q}]}(\texttt{student})(\texttt{leave})$
$= \lambda Q.\ \neg\underline{Q}(\texttt{student})(\texttt{leave})$
$= \lambda Q.\ \neg\underline{Q}(\texttt{leave})$

$[\![$every$_1$ student$]\!]^s$ can then saturate the Q argument:

(62) $[\![$every$_1$ student λ_1' not every$_1$ student leave$]\!]^s$
$= [\lambda Q.\ \neg Q(\texttt{leave})](\lambda P.\ [\texttt{EVERY}]_1^s(\texttt{student})(P))$
$= \neg[\texttt{EVERY}]_1^s(\texttt{student})(\texttt{leave})$

Naturally, when we evaluate with respect to `Stay`, we get the desired inverse scope interpretation:

(63) $[\![$every$_1$ student λ_1' not every$_1$ student leave$]\!]^{\texttt{Stay}} = \neg\texttt{EVERY}(\texttt{student})(\texttt{leave})$

Thus, the higher-type lambda abstraction required in order to perform semantic reconstruction is perfectly compatible with the analysis adopted in this paper.[12]

4.3 Lower-type swap states and ILRH

In the analysis adopted here, swap states trade in objects of type $(et)(et)t$, i.e., etetts. However, a reviewer notes that this is not necessary in order for the semantics to work: swap states could just as easily trade in lower-type, $(et)t$-type quantifiers. Using o as a variable over such lower-type swap states and adopting the same abbreviations as before, $[\![$every$_n]\!]^o$ can be defined as follows:

(64) $[\![$every$_n]\!]^o = \lambda P \lambda P'.\ [\texttt{EVERY}(P)]_n^o(P')$

Since `EVERY` is an etett, $\texttt{EVERY}(P)$ is of type $(et)t$, and can thus serve as an input to our new lower-type swap states. Now consider the following simple structure:

(65) every$_1$ student λ_1 every$_1$ student left

12 Note that if semantic reconstruction is permitted, it must be closely regulated. Otherwise, Fox & Nissenbaum's (1999) account of Williams's Generalization could be undone by inserting λ_n' instead of λ_n after the post-late merge QR, allowing the DP to scope below the extraposition site. This problem is not specific to my analysis and extends to any approach that permits semantic reconstruction.

Up until lambda abstraction, we get the following:

(66) $[\![\text{every}_1 \text{ student left}]\!]^o = [\texttt{EVERY}(\texttt{student})]^o_1(\texttt{leave})$

This takes us to lambda abstraction, which manipulates the lower-type swap state. Because of the lower types, rather than swapping in the etett $\texttt{THE}_z$ (where z is the lambda-abstracted variable), we swap in z's $(et)t$-type Montagovian type-lift, $\lambda P.\, P(z)$:

(67) $[\![\lambda_n \text{ X}]\!]^o = \lambda z.\, [\![\text{X}]\!]^{o[n,\ \lambda P.\ P(z)]}$

This gives us the following post-lambda-abstraction interpretation:

(68) $[\![\lambda_1 \text{ every}_1 \text{ student left}]\!]^o$
$= \lambda z.\, [\![\text{every}_1 \text{ student left}]\!]^{o[1,\ \lambda P.\ P(z)]}$
$= \lambda z.\, [\texttt{EVERY}(\texttt{student})]^{o[1,\ \lambda P.\ P(z)]}_1(\texttt{leave})$
$= \lambda z.\, [\lambda P.\, P(z)](\texttt{leave})$
$= \lambda z.\, \texttt{leave}(z)$

We thus have an et-type predicate that can recompose with $[\![\text{every}_1 \text{ student}]\!]$, and semantic composition can proceed as normal.

But notice that this no longer satisfies ILRH, a claim for which we saw empirical evidence in Section 2. The restrictor `student` makes no semantic impact at lower merge sites: its semantic contribution is wiped out along with the determiner's through lambda abstraction. This can, however, be prevented by enforcing ILRH in the lexical semantics of *every* and other scope-taking heads:

(69) $[\![\text{every}_n]\!]^o = \lambda P \lambda P'.\, [\texttt{EVERY}(P)]^o_n(\lambda x : P(x).\, P'(x))$

This gives us the following pre- and post-lambda-abstraction interpretations:

(70) a. $[\![\text{every}_1 \text{ student left}]\!]^o = [\texttt{EVERY}(\texttt{student})]^o_1(\lambda x : \texttt{student}(x).\, \texttt{leave}(x))$
b. $[\![\lambda_1 \text{ every}_1 \text{ student left}]\!]^o$
$= \lambda z.\, [\lambda P.\, P(z)](\lambda x : \texttt{student}(x).\, \texttt{leave}(x))$
$= \lambda z : \texttt{student}(z).\, \texttt{leave}(z)$

But this of course comes at a cost in the form of stipulation: why is the denotation of *every* (69) and not the seemingly simpler (64), and likewise for other determiners? Meanwhile, when using higher-type swap states the picture is simpler: for any determiner D there is an etett E such that $[\![\text{D}_n]\!]^s = [E]^s_n$.

It is worth emphasizing that in spite of invoking the empirical claims of ILRH, this is not an empirical argument in favor of using higher-type swap states: by using the somewhat more cumbersome definition in (69), ILRH is indeed still respected. I thus leave the choice between higher- and lower-type swap states as an open issue. However, regardless of one's preference between lower- and higher-type swap states, the fundamental point of the proposal adopted here remains the same: swap states of

some sort are a useful apparatus for compositionally deriving the semantic results of trace conversion.[13] With this in mind, I will continue to use higher-type swap states for the rest of this paper. We next turn to the task of extending our approach beyond DP quantification to include other types of scope-takers.

5 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 5.1 with the formal extension, revising the mechanism in a way that will permit lambda abstraction over arbitrary types. In Section 5.2 the power of this type-generalized formalism is directed toward an analysis of modals, first operating under the assumption that modals have syntactically represented restrictors similar to determiner quantifiers, then showing that the theory is powerful enough as is to allow ourselves to eschew this assumption and treat modals as lacking syntactically represented restrictors. Finally, in Section 5.3 the analysis is extended to account for degree phrase scope-taking in comparatives.

5.1 Type-generalized state dependency

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

(36) 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.

13 A reviewer notes that if not suitably constrained, swap states could potentially be powerful enough to define lexical items like *faux* and *nonce* from Section 3.2, something that was previously deemed problematic under certain views of compositionality. This is prevented on my analysis by making swap states a parameter of interpretation that can only be manipulated in very specific ways by lambda-abstractors: other heads can be sensitive to swap states, but cannot manipulate them in the ways required to define *faux* and *nonce*. It is worth noting that traditional variable assignments have to be similarly grammatically constrained, as they too are formally powerful devices that could be abused to create non-existent semantic interpretations. Thus, the need to constrain formally powerful devices in order to avoid non-existent interpretations is not specific to swap states.

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. 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:

(71) 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 (72). 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 $\lambda_{e,2}$.

(72) 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., $[\![\mathrm{every}_n]\!]^r = [\textsf{EVERY}]^r_n$.

In the previous section, our definition of lambda abstraction made use of the etett $\textsf{THE}_x$, 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, $\textsf{THE}_k$ will be type $(\alpha t)(\alpha t)t$. Luckily this is easy to define:

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

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

(74) 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:

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

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:

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

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.

5.2 Modals

I will use the sentences in (26), repeated below, to illustrate the analysis of modal scope:

(26) 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 (77); 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* syntactically reconstructing to its pre-movement position. As mentioned in the beginning of this section, I start by 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). I will later show 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.[14]

(77) 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:

(78) $[\![\text{Rivka leave the party}]\!]^{c,u} = 1$ 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):

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

(80) $[\![\text{INT Rivka leave the party}]\!]^{c,u} = \lambda v.$ 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 multiple-merge composition, we thus might define $[\![\text{can}]\!]^{c,u}$ as in (81), where 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:

(81) $[\![\text{can}]\!]^{c,u}_{\text{prelim}} = \mathsf{CAN}^c_u = \lambda p \lambda q.\ \mathsf{CAN}^c_u(p)(q)$

Note that it does not matter for our purposes what 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 CAN^c_u is of type $(wt)(wt)t$.

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: $[\![\text{can}]\!]$ must be state-sensitive in much the same way that $[\![\text{a}]\!]$ is. Since we have type-generalized our semantics by replacing swap states with state clusters, this is a simple matter:

14 I leave for future work the issue of how the present analysis might be integrated with an alternate theory in which possible worlds enter the compositional semantics through pronouns bound by lambda operators (Percus 2000). I see no reason to believe that any conflict should arise here.

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

Since $\texttt{CAN}^c_u$ is of type $(wt)(wt)t$, $[\texttt{CAN}^c_u]^r_n$ will be the $(wt)(wt)t$-type world-quantifier that $r\langle w\rangle$ swaps in for $\texttt{CAN}^c_u$ 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:

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

This then composes with $[\![\textsc{int}$ Rivka leave the party$]\!]^{c,u,r}$, which I will abbreviate as the proposition `rleave`:

(84) $[\![\text{can}_1\ \textsc{res}]\!]^{c,u,r}([\![\textsc{int}$ Rivka leave the party$]\!]^{c,u,r}) = [\texttt{CAN}^c_u]^r_1(R^c)(\texttt{rleave})$

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

(85) $[\![\text{not}]\!]^{c,u,r}([\![\text{can}_1\ \textsc{res int}$ Rivka leave the party$]\!]^{c,u,r})$
$= \neg[\texttt{CAN}^c_u]^r_1(R^c)(\texttt{rleave})$

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

(86) $[\![\text{not can}_1\ \textsc{res int}$ Rivka leave the party$]\!]^{c,u,\texttt{Stay}} = \neg\texttt{CAN}^c_u(R^c)(\texttt{rleave})$

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 (77b), in which *must* replaces *can* and takes scope over negation:

(77b) [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`:

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

Up to and including negation, the derivation for (77b) runs precisely parallel to that for (77a), leading to the following result:

(88) $[\![\text{not must}_1\ \textsc{res int}$ Rivka leave the party$]\!]^{c,u,r} = \neg[\texttt{MUST}^c_u]^r_1(R^c)(\texttt{rleave})$

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

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

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 $[\![\text{must}_1\ \textsc{res}]\!]$:

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

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

(91) $[\![\text{must}_1\ \textsc{res}\ \lambda_{w,1}\ \text{not must}_1\ \textsc{res}\ \textsc{int}\ \text{Rivka leave the party}]\!]^{c,u,\texttt{Stay}}$
$= \texttt{MUST}^c_u(R^c)(\lambda v : R^c(v).\ \neg\texttt{rleave}(v))$

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

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 (26b), rather than being (77b), would instead be the simpler (92):

(77b) $[\text{must}_1\ \textsc{res}]\ \lambda_{w,1}\ \text{not}\ [\text{must}_1\ \textsc{res}]\ \textsc{int}\ \text{Rivka leave the party}$

(92) $\text{must}_1\ \lambda_{w,1}\ \text{not must}_1\ \textsc{int}\ \text{Rivka leave the party}$

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? One possibility is to posit that in addition to the $(\alpha t)(\alpha t)t$-type swap state clusters used thus far, the semantics can also make use of lower-type swap state clusters of the sort briefly discussed in Section 4. While this is certainly doable, it is unnecessary: quantificational operators without syntactic restrictors can already be accounted for without any revisions to the theory at hand.

Suppose that on a traditional, non-multiple-merge semantics, the denotation for $[\![\text{must}]\!]^{c,u}$ is $\texttt{MST}^c_u$, which unlike $\texttt{MUST}^c_u$ takes a single propositional argument, making it type $(wt)t$. Recall that for any J of type $\alpha\beta$, $\underline{J} := \lambda K_\alpha.\ J$, i.e., J with a vacuous first argument tacked on. Thus, $\underline{\texttt{MST}^c_u}$ is a $(wt)(wt)t$-type quantifier $(\lambda p \lambda q.\ \texttt{MST}^c_u(q))$, meaning that it 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:

(93) $[\![\text{must}_n]\!]^{c,u,r} = \lambda q.\ [\underline{\texttt{MST}^c_u}]^r_n(\lambda v.\ 1)(q)$

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 (92). As before, the denotation up to and including the world-abstracting head INT is the wt-type proposition `rleave`. This is now fed to $[\![\text{must}_1]\!]$, which is of type $(wt)t$, giving us a truth value:

(94) $[\![\text{must}_1]\!]^{c,u,r}([\![\text{INT Rivka leave the party}]\!]^{c,u,r}) = [\underline{\texttt{MST}^c_u}]^r_1(\lambda v.\ 1)(\texttt{rleave})$

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

(95) $[\![\text{not}]\!]^{c,u,r}([\![\text{must}_1\ \text{INT Rivka leave the party}]\!]^{c,u,r}) = \neg[\underline{\texttt{MST}^c_u}]^r_1(\lambda v.\ 1)(\texttt{rleave})$

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

(96) $[\![\lambda_{w,1}\ \text{not must}_1\ \text{INT Rivka leave the party}]\!]^{c,u,r}$
$= \lambda v.\ [\![\text{not must}_1\ \text{INT Rivka leave the party}]\!]^{c,u,r[1,\texttt{THE}_v]}$
$= \lambda v.\ \neg[\underline{\texttt{MST}^c_u}]^{r[1,\texttt{THE}_v]}_1(\lambda v'.\ 1)(\texttt{rleave})$
$= \lambda v.\ \neg\texttt{THE}_v(\lambda v'.\ 1)(\texttt{rleave})$
$= \lambda v : [\lambda v'.\ 1](v).\ \neg\texttt{rleave}(v)$
$= \lambda v.\ \neg\texttt{rleave}(v)$

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

(97) $[\![\text{must}_1]\!]^{c,u,r}([\![\lambda_{w,1}\ \text{not must}_1\ \text{INT Rivka leave the party}]\!]^{c,u,r}) = 1$ iff
$[\underline{\texttt{MST}^c_u}]^r_1(\lambda v.\ 1)(\lambda v.\ \neg\texttt{rleave}(v))$

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

(98) $[\![\text{must}_1\ \lambda_{w,1}\ \text{not must}_1\ \text{INT Rivka leave the party}]\!]^{c,u,\texttt{Stay}}$
$= 1$ iff $\underline{\texttt{MST}^c_u}(\lambda v.\ 1)(\lambda v.\ \neg\texttt{rleave}(v))$
$= 1$ iff $\texttt{MST}^c_u(\lambda v.\ \neg\texttt{rleave}(v))$

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 higher-type swap states were equally effective in allowing for direct composition.

5.3 Degree phrases in comparatives

Finally we turn to comparatives, for which I will develop an account based on that of Bhatt & Pancheva (2004). Before diving into this analysis, it will help to discuss classical accounts of comparatives in the tradition of von Stechow (1984) and Heim (1985, 2000). I will use (99) as a sample sentence:

(99) Jo is taller than Al is.

On traditional accounts (99) has an LF along the lines of (100), in which the degree phrase headed by *-er* undergoes QR:

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

$[\![\text{tall}]\!]$ is a relation between a degree d and individual x, true iff x is at least d-tall:

(101) $[\![\text{tall}]\!]_{\text{trad.}} = \lambda d \lambda x.\ \texttt{height}(x) \geq d$

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. $[\![\text{-er}]\!]$ is thus of type $(dt)(dt)t$, i.e., a degree quantifier.

(102) $[\![(99)]\!]_{\text{trad.}} = [\![\text{-er}]\!](\lambda d.\ \texttt{height}(\texttt{al}) \geq d)(\lambda d.\ \texttt{height}(\texttt{jo}) \geq d)$

Two common denotations for *-er* that generate the correct truth conditions for (99) are provided in (103).

(103) a. $[\![\text{-er}]\!]_{\text{take 1}} = \lambda D \lambda D'.\ \texttt{max}(D) < \texttt{max}(D')$
b. $[\![\text{-er}]\!]_{\text{take 2}} = \lambda D \lambda D'.\ \exists d[\neg D(d) \wedge D'(d)]$

By using (103a), 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.

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

Using (103b), meanwhile, 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.

(105) $[\![(99)]\!]_{\text{take 2}} = 1$ iff $\exists d[\texttt{height}(\texttt{al}) \ngeq d \wedge \texttt{height}(\texttt{jo}) \geq d]$

Notice that on the traditional analysis, clausal comparatives constitute a form of antecedent-contained deletion. After all, the elided AdjP is contained within its own antecedent—the matrix AdjP, *taller than Al is*—and so a quantificational constituent headed by *-er* must undergo QR to avoid an infinite regress. The parallels between "normal" ACD and comparative ACD are illustrated in (106); in the former, the antecedent and elided phrases are VPs and the quantifier that undergoes movement is a DP, while in the latter the antecedent and elided phrases are AdjPs and the quantifier that undergoes movement is a DegP.

(106) a. Lisa [$_{\text{VP}}$ read [$_{\text{DP}}$ every book that Anna did [$_{\text{VP}}$ read]]]

b. Jo is [$_{\text{AdjP}}$ tall [$_{\text{DegP}}$ -er than Al is [$_{\text{AdjP}}$ tall]]]

We have already seen one theory of ACD in a multiple-merge theory of movement: namely that of Fox (2002), who argues that ACD involves (often string-vacuous) extraposition, with the relative clause late-merging with the noun it modifies after the DP has undergone (rightward) QR. As a result, ACD is a misleading name, as the ellipsis site is never actually contained within its antecedent. In fact, Bhatt & Pancheva (2004) argue at length that comparatives should be analyzed in a similar fashion, with the *than*-clause being extraposed by late-merging with *-er* after the latter has undergone (rightward) QR. I will not go through Bhatt & Pancheva's extensive arguments for this parallel here, and will instead focus on illustrating their theory and showing how it can be translated into the present analysis.

Bhatt & Pancheva (2004) set aside some of the finer details of the syntax-semantics of comparatives that are not directly relevant to their claims, so I will discuss their theory in broad strokes, and will then go into those finer details later when discussing my own version of their analysis. Roughly in keeping with Fox's (2002) account of ACD (but see below), Bhatt & Pancheva posit that first *-er* merges with (matrix) *tall* on its own, leading to the matrix clause *Jo is tall -er*. Then *-er* undergoes rightward QR, after which its *than*-clause complement merges with it, leading to a structure along the lines of (107):

(107) [Jo is tall [-er]] [-er than Al is tall]

After trace conversion, the LF structure that is fed to semantic interpretation is roughly as in (108). Note that they adopt a version of trace conversion that is slightly different from the one discussed earlier, but the semantic result is essentially the same.

(108) [λd_1 Jo is tall the d_1] [-er λd_2 than Al is tall the d_2]

This structure derives the correct interpretation: just like on the traditional account, the restrictor of *-er* is the set of height degrees not exceeding Al's height, and the scope of *-er* is the set of height degrees not exceeding Jo's height.

However, there is an important way in which the analyses of Bhatt & Pancheva (2004) and Fox (2002) diverge. For Fox, the quantifier and its restrictor merge as a unit in the pre-movement position, while the relative clause late merges with the restrictor, thereby intersectively modifying it at the higher merge site. Meanwhile, for Bhatt & Pancheva the quantifier *-er* is merged *on its own* at the lower site, with the restrictor itself being late merged after movement. A schematic representation of this distinction can be seen in (109):

(109) a. [$_{\text{XP}}$ … [Quant$_n$ Res] …] … [Quant$_n$ [Res ***Mod***]]
b. [$_{\text{XP}}$ … [Quant$_n$] …] … [Quant$_n$ ***Res***]

Since Bhatt & Pancheva's (2004) analysis does not give *-er* a restrictor at its lowest merge site, in order to avoid a type clash it is necessary that the version of trace conversion they adopt lend it one. But we are no longer using trace conversion, which means that something else must be done in order to make sure that *-er* is semantically restricted at its lowest merge site.[15]

In fact, Bhatt & Pancheva's (2004) analysis of comparatives is not the only proposal in which structures like (109b) crop up: Takahashi & Hulsey (2009) argue that such configurations also arise in the DP domain (see also Stanton 2016). To see why, note first that A′-movement often does not bleed Condition C when the R-expression (here, *John*) is contained within a nominal argument:

(110) a. * He$_i$ was sitting in the sunny corner of John$_i$'s room.
b. * Which corner of John$_i$'s room was he$_i$ sitting in?
(Takahashi & Hulsey 2009: p. 391, attributed to David Pesetsky)

This can be attributed to multiple-merge: in (110b), *John* sits both above and below the co-indexed *he*, and the lower position triggers a Condition C violation.

(111) * [which$_1$ corner of John$_i$'s room] was **he**$_i$ sitting in [which$_1$ corner of **John**$_i$'s room]

However, A-movement does seem to bleed Condition C in parallel configurations:

(112) a. * It seems to him$_i$ that every corner of John$_i$'s room is messy.
b. Every corner of John$_i$'s room seems to him$_i$ to be messy.

To account for this, Takahashi & Hulsey (2009) propose that with A-movement the entire NP *corner of John's room* can be *wholesale late merged* after movement has taken place, leading to the structure in (113). Independent factors prevent this from happening with A′-movement, hence why only A-movement bleeds Condition C.

(113) [every$_1$ corner of John$_i$'s room] seems to him$_i$ [every$_1$] to be messy

This structure is of the same sort as that proposed by Bhatt & Pancheva (2004) for comparatives: rather than a modifier of the restrictor being late merged after movement, it is the whole restrictor that is late merged. And much like Bhatt & Pancheva, Takahashi & Hulsey (2009) adopt a version of trace conversion that lends restrictors to restrictor-less lower copies, a path that is not available to us.

So without trace conversion, how can restrictors be given to unrestricted lower copies? One path forward is to use a type-shift that lends a vacuous restrictor to unrestricted quantifiers:

15 A reviewer suggests that this may be resolved by assigning *-er* a lower-type $((dt)t)$ denotation, obviating a *than*-clause restrictor at the lower merge site. But in this case it is not clear how the *than*-clause can then restrict *-er* after late merge does occur. In other words, the problem is not the need for lower *-er* to be interpreted as restrictor-less, but rather the mismatch between the required argument structures for higher and lower *-er* (restricted vs. restrictor-less, respectively).

(114) For J of type $(\alpha t)\beta$, $J^{\circ} := J(\lambda k_{\alpha}.\ 1)$

In this case, unrestricted ⟦-er⟧ or ⟦every⟧ could first undergo this type-shift before composing with its sister, while the syntactically restricted copies compose as normal. Another possibility would be to posit a rule of semantic interpretation that I will call Restriction-Inserted Composition (RIC), which has the same effect:[16]

(115) Restriction-Inserted Composition (RIC):
If $[\![\text{XP}]\!]^r$ is type $(\alpha t)(\alpha t)t$, then $[\![\text{Y XP}]\!]^r$ (= $[\![\text{XP Y}]\!]^r$) is the result of composing $[\![\text{Y}]\!]^r$ with $[\![\text{XP}]\!]^r(\lambda k_{\alpha}.\ 1)$.

Notice that RIC applies specifically to *phrase-level* $(\alpha t)(\alpha t)t$-type quantifiers: if a quantificational head goes unrestricted by a complement or specifier, a vacuously true restriction is given to it. This way, *-er* can semantically compose without a syntactic restrictor at its lowest merge site through RIC—at the lowest merge site *-er* is its own DegP—and with a restrictor at its highest merge site through normal function application. That is, something like the structure in (116) is interpretable:

(116) [$\lambda_{d,1}$ Jo is tall er_1] [er_1 than Al is tall]

While I will not go through the details, this works for wholesale late merger structures like (113) as well, since at its lower merge site *every* is a one-node maximal projection, and can thus compose with ⟦to be messy⟧ through RIC. The choice between the type-shift J° and RIC is an interesting one; I adopt RIC for concreteness, but leave the choice between the two as an open issue.

Before going through the full compositional semantics for comparatives, one more thing needs to be said on the syntactic side of things: namely, how the internal argument of the *than*-clause's *tall* is saturated. While there are multiple ways of accomplishing this, I will opt for an approach that takes direct inspiration from the matching theory of relative clauses discussed earlier. Suppose that the *than*-clause-internal *tall* has its internal argument saturated not by an operator as in traditional accounts, but by another *-er*. This *-er* then undergoes *wh*-movement triggering lambda abstraction, and is deleted upon matching with the instance of *-er* with which the *than*-clause late merges. This is illustrated in (117):

(117) [$\lambda_{d,1}$ Jo is tall er_1] [er_1 [$_{\text{CP}}$ ~~er_2~~ $\lambda_{d,2}$ than Al is tall er_2]]

In order for this to work, it must be the case that not only is the higher er_2 deleted at PF, but it is also deleted—or at least bleached of all semantic content—at LF. Otherwise, there will eventually be a type clash.[17] It is worth noting that on my analysis,

16 Note that if the semantics is to be deterministic, either RIC or normal function application must be assigned precedent when both are possible, e.g., if $[\![\text{XP}]\!]^r$ is type $(\alpha t)(\alpha t)t$, and $[\![\text{Y}]\!]^r$ is type αt.

17 More specifically, the *than*-clause will have an interpretation that is either of type t (if er_2 and the lambda-abstracted constituent compose by RIC) or of type $(dt)t$ (if they compose via normal function application). Neither type is able to compose with $(dt)(dt)t$-type er_1.

the same must be said for any operator whose semantic contribution is exclusively to trigger lambda abstraction. After all, that operator must have a well-defined denotation in order to compose at the lower merge site, but it must not make its contribution at the higher merge site and thereby "undo" the lambda abstraction it has triggered. Thus, the higher instance of the operator must be either semantically bleached or removed entirely. This extends equally well to matching theories of relative clauses. Take, for example, the NP *book that I like*:

(118) [$_{\text{NP}}$ book [$_{\text{CP}}$ ~~Op_1 book~~ λ_1 that I like Op_1 book]]

Clearly in order for composition to work within the relative clause, $[\![\text{Op}_1]\!]^r$ must be an etett. But then the result of lambda abstraction must not be allowed to recompose with Op_1 *book*, or else we will end up with something of type t, which cannot compose with *book*. Thus, the higher Op_1 *book* must be stripped of semantic content or removed from the LF structure entirely after matching the two instances of *book*: in some form, deletion upon matching must apply equally well to both PF and LF.

For convenience, I will assume that the higher er_2 is well and truly deleted after matching, rather than just semantically bleached. With this in mind, the final interpreted structure for (99) will be as in (119):

(119) [$\lambda_{d,1}$ Jo is tall er_1] [er_1 [$_{\text{CP}}$ $\lambda_{d,2}$ than Al is tall er_2]]

Our next step is to determine how this syntactic structure actually semantically composes. We start with *-er*, which I assign the following denotation, where ER is some $(dt)(dt)t$-type degree-quantifier; I will assume it is one of the definitions in (103):

(120) $[\![\text{-er}_n]\!]^r = [\texttt{ER}]^r_n \qquad (= \lambda D \lambda D'.\ [\texttt{ER}]^r_n(D)(D'))$

Let us start in the *than*-clause, meaning that we compose $[\![\text{-er}_2]\!]$ with $[\![\text{tall}]\!]$. Previously, $[\![\text{tall}]\!]$ was taken to denote a relation between degrees and individuals. But in the same way that verbs like $[\![\text{like}]\!]$ had to be assigned higher types in order to take $(et)t$-type quantificational arguments, $[\![\text{tall}]\!]$ will instead be assigned a higher type in order to take $(dt)t$-type arguments:

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

We can now compose $[\![\text{-er}_2]\!]$ and $[\![\text{tall}]\!]$. Since the former is type $(dt)(dt)t$, while the latter is looking for a $(dt)t$-type argument, they cannot compose via normal function application. However, since *-er*$_2$ is a phrase (DegP), this configuration is eligible for composition via RIC. The result is as follows:

(122) $[\![\text{tall -er}_2]\!]^r$

$= [\![\text{tall}]\!]^r([\![\text{-er}_2]\!]^r(\lambda d.\ 1))$ (RIC)

$= [\![\text{tall}]\!]^r([\texttt{ER}]^r_2(\lambda d.\ 1))$

$= [\lambda G \lambda x.\ G(\lambda d.\ \texttt{height}(x) \geq d)]([\texttt{ER}]^r_2(\lambda d.\ 1))$

$= \lambda x.\ [\texttt{ER}]^r_2(\lambda d.\ 1)(\lambda d'.\ \texttt{height}(x) \geq d')$

This then composes with $[\![\text{Al}]\!]^r$, which I simply take to be the *e*-type `al`, though this can be easily revised if one wishes to allow names to take scope:

(123) $[\![\text{Al is tall -er}_2]\!]^r = [\texttt{ER}]^r_2(\lambda d.\ 1)(\lambda d'.\ \texttt{height}(\texttt{al}) \geq d')$

Continuing to assume that *than* is semantically vacuous, next we lambda abstract:

(124) $[\![\lambda_{d,2}$ than Al is tall $\text{er}_2]\!]^r$
$= \lambda d.\ [\![\text{than Al is tall er}_2]\!]^{r[2,\texttt{THE}_d]}$
$= \lambda d.\ [\texttt{ER}]^{r[2,\texttt{THE}_d]}_2(\lambda d'.\ 1)(\lambda d''.\ \texttt{height}(\texttt{al}) \geq d'')$
$= \lambda d.\ \texttt{THE}_d(\lambda d'.\ 1)(\lambda d''.\ \texttt{height}(\texttt{al}) \geq d'')$
$= \lambda d.\ \texttt{height}(\texttt{al}) \geq d$

The result is that the *-er* that does the actual degree-quantification—the higher merge site of *-er*$_1$—has precisely the same restrictor as in traditional analyses: the set of degrees not exceeding Al's height.

The restrictor of the quantifying *-er*$_1$ (i.e., the matrix clause up to and including lambda abstraction) composes in precisely the same way: *tall* and *-er*$_1$ compose via RIC, with *Jo* saturating the entity argument, followed by lambda abstraction over index 1. The result is the set of degrees not exceeding Jo's height, again exactly as in traditional theories:

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

The restrictor and scope of highest *-er*$_1$ then compose with it in turn, generating the following interpretation for the LF in (119):

(126) $[\![[\lambda_{d,1}$ Jo is tall $\text{er}_1]$ $[\text{er}_1\ \lambda_{d,2}$ than Al is tall $\text{er}_r]]\!]^r$
$= [\texttt{ER}]^r_1(\lambda d.\ \texttt{height}(\texttt{al}) \geq d)(\lambda d.\ \texttt{height}(\texttt{jo}) \geq d)$

And when evaluating with `Stay`, we get the following finalized interpretation:

(127) $[\![[\lambda_{d,1}$ Jo is tall $\text{er}_1]$ $[\text{er}_1\ \lambda_{d,2}$ than Al is tall $\text{er}_r]]\!]^{\texttt{Stay}}$
$= \texttt{ER}(\lambda d.\ \texttt{height}(\texttt{al}) \geq d)(\lambda d.\ \texttt{height}(\texttt{jo}) \geq d)$

What these truth conditions end up being depends on how one defines `ER`. If we use (103a) we get (104), and if we use (103b) we get (105). Either way, the desired truth conditions obtain: namely, the same ones derived on traditional analyses.

Summing up, we have seen that Bhatt & Pancheva's (2004) analysis of comparatives can be translated into the theory proposed in this paper, so long as something is done to ensure that the degree-quantifier is semantically restricted at its lower merge sites. This can be accomplished with a type-shift or the composition rule RIC, either of which can also be used to interpret structures with wholesale late merger of the sort that has been argued to be a possibility in A-movement configurations.

6 Concluding remarks

In this paper I have outlined a form of *compositional trace conversion* that generates the desired semantic effects of trace conversion, but without the syntactic stipulations or compositional difficulties of its syntactic and semantic variants. This analysis was shown to extend beyond 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 most obvious issue is empirical coverage. While I have attempted to illustrate the broad applicability of my analysis 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. Note that while the interpretations of pronouns could perhaps be defined via swap states, it is also possible for swap state clusters to coexist with variable assignment clusters, with lambda-abstraction simultaneously manipulating swap states and variable assignments:

(128) $[\![\lambda_{\alpha,n}\ \mathrm{X}]\!]^{r,h} = \lambda k_{\alpha}.\ [\![\mathrm{X}]\!]^{r[n,\mathsf{THE}_k],\ h[n,k]}$

Thus, the theory in this paper is by all appearances fully compatible with a traditional approach to pronouns as denoting (free or bound) variables. However, further elucidation of these topics is 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:[18]

(129) Abstract and Apply (AA):
If $[\![\mathrm{X}_n]\!]^r$ is type $(\alpha t)t$, and $[\![\mathrm{Y}]\!]^r$ is type t, then
$[\![\mathrm{X}_n\ \mathrm{Y}]\!]^r = [\![\mathrm{X}_n]\!]^r(\lambda k_{\alpha}.\ [\![\mathrm{Y}]\!]^{r[n,\mathsf{THE}_k]})$

This rule seems to generate the right results for the basic cases, but questions may arise about its possible stipulativeness. This also leaves open the question of what to do about operators like *Op*, since on the present approach they trigger lambda abstraction but do not semantically compose at the higher merge site (see the discussion in Section 5.3). I leave these issues for future work.

18 If one wishes to permit semantic reconstruction then presumably there need to be two versions of AA—reconstructing and non-reconstructing—with optionality in determining which rule is applied.

Moreover, as a reviewer notes, another potential problem comes from the semantically interpreted syntactic indices that appear on scope-taking heads, which some have argued to themselves constitute violations of Inclusiveness (see, e.g., Chomsky 2000). My analysis, like many others, makes crucial use of these indices in the lexical semantics of quantificational heads and the interpretation of lambda abstraction. Perhaps the most promising means of obviating syntactic indices is not to eliminate indices altogether, but rather to treat them as objects that are assigned through compositional semantic mechanisms, rather than in the syntax. But this, again, is a matter that must be left for later exploration.

References

Baltin, Mark. 1987. Do antecedent-contained deletions exist? *Linguistic Inquiry* 18(4). 579–595.

Barwise, Jon & Robin Cooper. 1981. Generalized quantifiers and natural language. *Linguistics and Philosophy* 4(2). 159–219.

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

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

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

Chomsky, Noam. 2000. Minimalist inquiries: The framework. In Roger Martin, David Michaels & Juan Uriagareka (eds.), *Step by step: Essays on Minimalist syntax in honor of Howard Lasnik*, 89–155. Cambridge, MA: MIT Press.

Cresti, Diana. 1995. Extraction and reconstruction. *Natural Language Semantics* 3(1). 79–122.

Erlewine, Michael Yoshitaka. 2014. *Movement out of focus*. Cambridge, MA: MIT dissertation.

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

Fox, Danny. 1999. Focus, parallelism, and accommodation. In Tanya Matthews & Devon Strolovitch (eds.), *Semantics and linguistic theory (SALT)*, vol. 18, 70–90. Washington, D.C.: Linguistic Society of America.

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.

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

Gotham, Matthew. 2018. Making Logical Form type-logical: Glue semantics for Minimalist syntax. *Linguistics and Philosophy* 41(5). 511–556.

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.

Kennedy, Christopher. 1994. Argument contained ellipsis. Linguistics Research Center Report LRC-94-03. University of California, Santa Cruz.

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

Larson, Richard K. & Robert May. 1990. Antecedent containment or vacuous movement: Reply to Baltin. *Linguistic Inquiry* 21(1). 103–122.

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: Uni-

versity of Massachusetts GLSA.
Lechner, Winfried. 1998. Two kinds of reconstruction. *Studia Linguistica* 52(3). 276–310.
May, Robert. 1977. *The grammar of quantification*. Cambridge, MA: MIT dissertation.
May, Robert. 1985. *Logical Form: Its structure and derivation*. Cambridge, MA: MIT Press.
Moulton, Keir. 2015. CPs: Copies and compositionality. *Linguistic Inquiry* 46(2). 305–342.
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.
Rooth, Mats. 1992. Ellipsis redundancy and reduction redundancy. In Steve Berman & Arild Hestvik (eds.), *Proceedings of the Stuttgart Ellipsis Workshop*, Heidelberg: IBM Germany.
Rullmann, Hotze. 1995. *Maximality in the semantics of wh-constructions*. Amherst, MA: University of Massachusetts Amherst dissertation.
Ruys, E.G. 2015. A Minimalist condition on semantic reconstruction. *Linguistic Inquiry* 46(3). 453–488.
Sag, Ivan. 1976. *Deletion and logical form*. Cambridge, MA: MIT dissertation.
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.
Sharvit, Yael. 1999. Connectivity in specificational sentences. *Natural Language Semantics* 7(3). 299–339.
Sportiche, Dominique. 2005. Division of labor between merge and move: Strict locality of selection and apparent reconstruction paradoxes. UCLA, Ms.
Stanton, Juliet. 2016. Wholesale late merger in A′-movement: Evidence from preposition stranding. *Linguistic Inquiry* 47(1). 89–126.
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.
von Stechow, Arnim. 1991. Syntax und semantik. In Arnim von Stechow & Dieter Wunderlich (eds.), *Semantik/semantics: An international handbook of contemporary research*, 90–148. Berlin: Walter de Gruyter.
Takahashi, Shoichi & Sarah Hulsey. 2009. Wholesale late merger: Beyond the A/A′ distinction. *Linguistic Inquiry* 40(3). 387–426.

Williams, Edwin. 1974. *Rule ordering in syntax*. Cambridge, MA: MIT dissertation.
Wurmbrand, Susi. 2010. Reconstructing the A/A′-distinction in reconstruction. In Jon Scott Stevens (ed.), *U. Penn Working Papers in Linguistics* 16(1). 245–254.

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