Composing copies without trace conversion*

Robert Pasternak
Leibniz-Center for General Linguistics (ZAS)
pasternakrs@gmail.com

July 16, 2019
Comments welcome and much appreciated

Abstract

The copy theory of movement presents well-known challenges for the interpretation of QR, as lower copies must be interpreted like bound variables, and higher copies like true quantifiers. A common solution to this problem is to utilize a (post-)syntactic operation of trace conversion that converts lower copies into bound definites (Fox 2002, 2003). This paper introduces a new theory that allows for the direct composition of copies without trace conversion, thereby avoiding certain problematic features of this operation. Instead, the semantic effects of trace conversion are automatically generated via lambda abstraction. The analysis is shown to work not only for DPs, but also for modals taking scope over negation and QR of degree phrases in comparatives.

1 Introduction

For much of the history of modern generative syntax since Chomsky 1957, displacement—the apparent tendency for constituents to simultaneously occupy multiple syntactic locations—has been cast in terms of movement: a constituent seems to occupy multiple locations because it starts in one spot and moves to another, leaving a trace. However, in pursuit of his Minimalist Program, Chomsky (1995) recasts displacement in terms of copies: a constituent occupies multiple locations by having a copy at each, with only one such copy generally being pronounced. While this copy theory of movement has proved useful in certain areas of syntactic inquiry and represents a desirable push toward minimizing syntactic machinery, it presents something of a puzzle for semantic interpretation. To see why, consider (1b), a

*For helpful discussion, many thanks to Patrick Elliott, Uli Sauerland, and audience members at the ZAS Semantics and Pragmatics Reading Group and the University of Göttingen’s Oberseminar English Linguistics. Special thanks to Patrick Elliott for many long and fruitful conversations on all aspects of this project.
plausible LF for an inverse scope interpretation of (1a) in the copy theory of movement.¹ This can be contrasted with (1c), a more traditional LF with traces.

(1)  
   a. A student likes every teacher.
   b. [TP [every₂ teacher] λ₂ [TP [a₁ student] λ₁ T [VP a₁ student like every₂ teacher]]]
   c. [TP [every₂ teacher] λ₂ [TP [a₁ student] λ₁ T [VP t₁ like t₂]]]

A common approach to interpreting LFs like (1c) is to treat traces as free variables that are bound by the lambda-abstracting nodes λ₁ and λ₂, generating predicates that serve as arguments to their respective quantificational DPs (Heim & Kratzer 1998). But this view of compositionality seems untenable in the face of an LF like (1b), as it would require that the DP every teacher, for example, be interpreted as a true quantifier at the higher copy, and as a bound variable (more or less) at the lower copy. Put another way, there is an apparent tension between the following principles of semantic interpretation: (i) the highest copy of a quantificational DP is responsible for quantification; (ii) lower copies of a quantificational DP do not introduce quantification, and are interpreted as bound variables; and (iii) syntactically identical DPs are also semantically identical.

One way of dissolving this tension is to accept the above three principles and posit a post-syntactic operation that alters the structure of lower copy DPs in a way that leads to a bound variable-like interpretation. This way, (iii) is maintained but irrelevant, since higher and lower copies are no longer syntactically identical. For example, Fox (2002, 2003) proposes trace conversion, which converts lower copies of quantificational determiners into bound definites, more or less as in (2):²

(2)  
   [TP [every₂ teacher] λ₂ [TP [a₁ student] λ₁ T [VP the₁ student like the₂ teacher]]]

An alternate version of trace conversion mentioned by Fox (2003) states that it is not syntactic but semantic: the semantics interprets lower copies of quantificational DPs as if some syntactic alteration had taken place, and thus differently from their highest copy:

(3)  
   **Semantic Trace Conversion** (Fox 2003, p. 110):  
   In a structure formed by DP movement, DPₙ[ϕ…DPₙ…], the derived sister of DP, ϕ, is interpreted as a function that maps an individual, x, to the meaning of ϕ[x/ₙ].
   ϕ[x/ₙ] is the result of substituting every constituent with the index n in ϕ with himₓ, a pronoun that denotes the individual x.

Thus, unlike syntactic trace conversion, semantic trace conversion resolves the tension between the above three principles simply by abandoning (iii).

While trace conversion—usually in its syntactic variant—is the most popular means of interpreting copied quantifiers, it is not without its drawbacks, both theory-internal and

---

¹A note on indices is in order, since the proposed semantics relies on them. The indices in (1b) are not like those in work on, for example, pronouns: they do not indicate variable-binding or co-reference, but “co-copyhood”. As Chomsky (1995, p. 227) notes, some such marking is independently needed to distinguish between otherwise identical non-copies. See Collins & Stabler 2016 for similar discussion.

²For a different but thematically similar solution, see Sauerland 1998, 2004.
empirical. The main theory-internal issue with trace conversion in both its syntactic and semantic manifestations is that it is anti-compositional. Take, for instance, syntactic trace conversion. On this approach, trace conversion performs a syntactic operation only on lower copies of a DP. But when a DP is initially merged into the structure, without any recourse to look-ahead there is no way of telling whether that copy will be the highest copy (and thus not undergo trace conversion) or a lower copy. In other words, we do not know that a given copy of a DP is a lower copy until movement has already taken place, at which point the lower copy is already embedded in a larger structure. Moreover, syntactic trace conversion is accomplished by inserting lexical material absent from the numeration (the), in violation of Chomsky’s (1995) otherwise robust Inclusiveness Condition. Meanwhile, on the semantic approach to trace conversion this syntactic anti-compositionality is replaced with semantic anti-compositionality: the interpretation of a quantificational DP is not a function of the interpretations of its parts, since the same DP is interpreted like a pronoun in some syntactic environments and like a true quantifier in others.

On an empirical level, trace conversion—at least in its syntactic variant, with the relative success of the semantic variant depending on how precisely it is (re)formulated—suffers from the fact that DPs are not the only type of syntactic constituent that takes scope by means of QR. For instance, degree phrases in comparatives can give rise to scope ambiguities, as evidenced by examples (4) and (5) from Heim (2000, p. 48), 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:

(4) (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.  \(\text{require} > \text{DegP}\)
   b. The paper must be at least 15 pages.  \(\text{DegP} > \text{require}\)

(5) (This draft is 10 pages.) The paper is required to be less long than that.
   a. The paper must be shorter than 10 pages.  \(\text{require} > \text{DegP}\)
   b. The minimum length is below 10 pages.  \(\text{DegP} > \text{require}\)

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 negation; modals like can that scope under negation then reconstruct to their pre-movement positions, while those like must that scope over negation are interpreted in their post-movement positions, leaving a trace—or, assuming the copy theory of movement, a copy—in their merge positions.

(6) a. Rivka cannot leave the party.  \(\text{LF: [not [can...]]}\)
   b. Rivka must not leave the party.  \(\text{LF: [must}_1 \lambda_1 [\text{not } [t_1...]]}\)

But of course, in order for syntactic trace conversion to take place, there needs to be a determiner that can be swapped out for bound the. Given that the scope-bearing quantifiers in (4–5) are the degree morphemes -er and less, and in (6b) the modal must, a syntactic process
replacing each of these with the determiner *the* at lower copies seems undesirable.3

An intriguing alternative to trace conversion, first proposed by Johnson (2012) and further developed by Fox & Johnson (2016), resolves the tension not by positing an operation modifying lower copies or by giving quantificational DPs different interpretations at different positions, but instead by jettisoning principle (i) above. In short, the claim is that instead of higher copies being quantificational and lower copies being non-quantificational, determiners are semantically interpreted as bound definites from the get-go, with a separate head contributing the quantification traditionally associated with quantificational determiners. QR, rather than involving the whole DP undergoing covert movement, instead involves only the restrictor moving (or “moving”) from the complement of the determiner to the restrictor of the true quantificational head. This is illustrated in (7), where *student* is the unpronounced higher copy of *student*, and ∀ is the head that provides the true universal quantification:4

\[
(7) \quad [\forall_1 \text{ student}] \lambda_1 \text{ Al likes every}_1 \text{ student}
\]

If [every1] = [the1] and [∀] is the traditional universal quantifier, the LF in (7) generates the desired truth conditions. It is worth noting that this is not the first time that the quantificational force traditionally associated with a determiner has been divorced from the semantics of the determiner itself, with perhaps the most famous example being Heim’s (1982) treatment of indefinites. Also worth noting is that this analysis elegantly avoids the problems of anti-compositonality faced by trace conversion: there is no need to syntactically modify lower copies, and semantic computation can proceed in a straightforwardly bottom-up fashion.

However, perhaps the greatest concern that one might have about Johnson’s analysis is its possible lack of universality. On a macro level, this includes concerns about generalization to degree phrases and modals: is -er in (4) really a bound degree-denoting definite, with the restrictor of the degree phrase undergoing QR to restrict the true (silent) degree quantifier? Is the combination of *must* plus its covert restrictor really a bound world-denoting definite description, with the actual world-quantifier farther up the tree? And on the micro level, in order for Johnson’s analysis to fully resolve the problem, it must work for all quantifiers, meaning that each needs its own separate “truly quantificational” head.

These are all, of course, perfectly bitable bullets, and likewise for both syntactic and semantic trace conversion. However, it does leave one wondering whether a bullet-free alternative is available. In this paper, I will illustrate what one such alternative could look like, providing a compositional semantics in which structures with multiple copies can be interpreted directly, without any anti-compositional syntactic or semantic operations. The trick is that a mechanism will be included in the compositional semantics that will allow for an operator’s traditional quantificational denotation to be “swapped out” for a bound variable-like denotation. The operation of lambda abstraction automatically triggers this swapping,

---

3Heim (2006a) argues that *less* is not monomorphemic, but is really composed of *er* + *little*, with *er* doing the degree quantification. In this case, for both (4) and (5) it is *er* that must be replaced with *the*.

4Rather than framing things in terms of copies, Johnson’s analysis uses the multidominance theory of movement. This distinction is important for him, but is immaterial to the concerns at hand.
allowing for immediate recombination with the quantificational constituent. In other words, lambda abstraction performs the work of trace conversion without any actual operation of trace conversion. In Section 2 I develop this analysis with quantificational DPs. But since “trace conversion” is on this analysis completely divorced from the syntactic category of determiner, there is no principled reason why this should not work with other scope-takers besides DPs. With this in mind, in Section 3 I show that this analysis extends equally well to modal and degree phrase scope-taking. Section 4 concludes.

2 Swap states and the composition of DP copies

In going over how the system works for DP quantifier interpretation, we will use the LF in (1b), repeated below, as our example:

(1b) \[ \text{TP [every}_2\text{ teacher]} \lambda_2 \text{TP [a}_1\text{ student]} \lambda_1 \text{ T [VP a}_1\text{ student like every}_2\text{ teacher]}}] \]

In order for our compositional semantics to work, we need some mechanism that will allow us to “swap out” a determiner’s quantificational interpretation for a bound variable-like interpretation at lower copies, but in a straightforwardly bottom-up compositional fashion. In order to do this I will make use of what I will call swap states, which I will sometimes just call states. 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.\(^5\) That makes a swap state the somewhat cumbersome type \( n((et)(et)t)((et)(et)t) \), which I will abbreviate as \( s \). I will also use \( s \) as a variable over swap states. For a given state \( s \), index \( n \), and etetts \( D \) and \( D' \), if \( s(n)(D) = D' \), I will say that \( s \) swaps out \( D \) for \( D' \) at (index) \( n \), or equivalently, \( s \) swaps in \( D' \) for \( D \) at (index) \( n \). For readability’s sake, I will rewrite \( s(n)(D) \) as \( Dsn \).

So now that we have swap states, how are they actually used? In short, they serve a role very much analogous to variable assignments in more traditional approaches like that of Heim & Kratzer (1998). Tradition has it that variable assignments are a parameter of semantic interpretation, and lambda abstraction generates 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 (8):

(8) Traditional Lambda Abstraction: (cf. Heim & Kratzer 1998)
\[
[\lambda_n X]g = \lambda x. [X]g[n,x],
\]
where \( g[n,x] \) is the \( g' \) such that \( g'(n) = x \) and for all \( m \neq n \), \( g'(m) = g(m) \).

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 important gaps to be filled in later, can be seen in (9):

\(^5\) 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 \( \langle\alpha\beta\rangle\gamma \) is what would traditionally be written as \( \langle\langle\alpha,\beta\rangle,\gamma\rangle \).
New Lambda Abstraction (Preview):
\[
[\lambda_n X]^s = \lambda x. [X]^{s[n,?]},
\]
where \(s[n,?]\) is the \(s'\) such that...

The plan is that whatever \([X]^{s[n,?]}\) looks like, it will serve as a semantic simulation 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 \textit{teacher} and \textit{student} are as one might expect: they are ett-type predicates. These can be seen in (10):

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

I will often rewrite “\(\lambda x. \text{teacher}(x)\)” as the metalanguage object “teacher” when convenient, and likewise for “\(\lambda x. \text{student}(x)\)”. As for \([\text{like}]^s\), this is again more or less as one would expect, except that the types must be shifted in order to allow \([\text{like}]^s\) to directly compose with two (et)\(t\)-type quantificational arguments. This leads to the definition in (11):

(11) \([\text{like}]^s = \lambda Q_{(et)} \lambda Q'_{(et)}. Q'(\lambda x. Q(\lambda y. \text{like}(x, y)))\)

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

(12) a. \([a_1]^s = \lambda P_{et} \lambda P'_{et}. \text{SOME}_1^s(P)(P')\)
b. \([\text{every}_2]^s = \lambda P_{et} \lambda P'_{et}. \text{EVERY}_2^s(P)(P')\)

Composing our VP involves relatively straightforward function application. First we combine \([\text{every}_2]^s\) with \([\text{teacher}]^s\), and then feed the result to \([\text{like}]^s\) as its first argument:

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

Next we combine \([a_1]^s\) with \([\text{student}]^s\) and feed the result to \([\text{like every}_2 \text{teacher}]^s\):

(14) a. \([a_1]^s([\text{student}]^s) = \lambda P'. \text{SOME}_1^s(\text{student})(P')\)
b. \([\text{like every}_2 \text{teacher}]^s([a_1 \text{ student}]^s) = 1 \text{ iff } [a_1 \text{ student}]^s(\lambda x. \text{EVERY}_2^s(\text{teacher})(\lambda y. \text{like}(x, y))), \text{i.e., }\)
   \([\text{SOME}_1^s(\text{student})(\lambda x. \text{EVERY}_2^s(\text{teacher})(\lambda y. \text{like}(x, y)))\]

6
And just like that, our VP is composed.

Since we are treating tense (T) as semantically vacuous for simplicity’s sake, the next step is lambda abstraction via $\lambda_1$. As mentioned above, this entails hijacking the swap state in order to swap out the quantificational etett for a bound variable etett. In order to do this, though, we must know what our bound variable etett is. Whatever it is, it must be parameterized to an individual: namely, the entity argument that is lambda-abstracted over. Three immediate options for this bound variable etett spring to mind, defined below (where $z$ is the lambda-abstracted entity argument):

\begin{align*}
(15) & \quad a. \quad BD_z := \lambda \lambda P \lambda P'. P(z) \land P'(z) \\
& \quad b. \quad BD'_z := \lambda \lambda P \lambda P'. P'(z) \\
& \quad c. \quad \text{THE}_z := \lambda \lambda P \lambda P' : P(z). P'(z)
\end{align*}

If $BD_z$ is swapped in for a quantificational etett, the prediction is that both the restrictor and the scope of a lower copy are interpreted, and both contribute assertive content. Meanwhile, if $BD'_z$ is swapped in for a quantificational etett, we predict that the restrictor of lower copies makes no semantic contribution whatsoever: $\lambda P$ binds nothing. Finally, $\text{THE}_z$, which would generate results identical to those of trace conversion as implemented by Fox (2002, 2003), is like $BD_z$ in that it treats the restrictors of lower copies as making semantic contributions, but is unlike $BD_z$ in that the restrictor is presuppositional rather than assertive.\footnote{For those familiar with Fox’s work, the equivalence between my $\text{THE}_z$ and Fox’s trace conversion may not be obvious. However, note that $\text{THE}_z$ is equivalent to $\lambda \lambda P \lambda P' : P'(z)[\lambda P(z) \land z]$}, a type-lifted version of the semantic result of Fox’s trace conversion.

If we have decided which bound variable etett to swap in for SOME when lambda abstracting over index 1, we next need to decide on how to actually perform this swap. In the traditional lambda abstraction in (8), 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 $s[1, \text{THE}_z]$, which is the state identical to $s$ except that $s[1, \text{THE}_z]$ swaps out all etetts for $\text{THE}_z$ at index 1. More generally:

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

As promised, $s[n, D]$ is identical to $s$ for all indices other than $n$, but at $n$ it swaps out all etetts for $D$. With this in place, we now have the formal tools necessary in order to define lambda abstraction and fill in the blanks in (9). This can be seen in (17):
Let's try out our new lambda abstraction on our test example. The result is as follows:

(18) \[
\begin{aligned}
\mathllbracket \lambda_1 a_1 \text{ student like every}_2 \text{ teacher} \mathrrbracket^s & = \lambda z. \mathllbracket a_1 \text{ student like every}_2 \text{ teacher} \mathrrbracket^s[1, \text{THE}_2] \\
& = \lambda z. \text{SOME}_1^s[1, \text{THE}_2](\text{student})(\lambda x. \text{EVERY}_2^s[1, \text{THE}_2](\text{teacher})(\lambda y. \text{like}(x, y)))
\end{aligned}
\]

By definition, for any state \(s\) and etett \(D\), \(D_1^s[1, \text{THE}_2] = \text{THE}_2\), meaning that \(\text{SOME}_1^s[1, \text{THE}_2]\) can be replaced with \(\text{THE}_2\). Similarly, for any state \(s\), etett \(D\), and \(n \neq 1\), \(D_n^s[1, \text{THE}_2] = D_n^s\), meaning that \(\text{EVERY}_2^s[1, \text{THE}_2]\) can be replaced with \(\text{EVERY}_2^s\).

(19) \[
\begin{aligned}
\mathllbracket \lambda_1 a_1 \text{ student like every}_2 \text{ teacher} \mathrrbracket^s & = \lambda z. \text{THE}_2(\text{student})(\lambda x. \text{EVERY}_2^s(\text{teacher})(\lambda y. \text{like}(x, y))) \\
& = \lambda z : \text{student}(z). \text{EVERY}_2^s(\text{teacher})(\lambda y. \text{like}(z, y))
\end{aligned}
\]

We now have an \(et\)-type predicate, which naturally can be fed back into \(\mathllbracket a_1 \text{ student} \mathrrbracket^s\):

(20) \[
\begin{aligned}
\mathllbracket a_1 \text{ student} \mathrrbracket^s(\mathllbracket \lambda_1 a_1 \text{ student like every}_2 \text{ teacher} \mathrrbracket^s) & = 1 \text{ iff} \\
\text{SOME}_1^s(\text{student})(\lambda z : \text{student}(z). \text{EVERY}_2^s(\text{teacher})(\lambda y. \text{like}(z, y)))
\end{aligned}
\]

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

(21) \[
\begin{aligned}
\mathllbracket \lambda_2 a_1 \text{ student } \lambda_1 a_1 \text{ student like every}_2 \text{ teacher} \mathrrbracket^s & = \lambda x. \mathllbracket a_1 \text{ student } \lambda_1 a_1 \text{ student like every}_2 \text{ teacher} \mathrrbracket^s[2, \text{THE}_2] \\
& = \lambda x. \text{SOME}_1^s[2, \text{THE}_2](\text{student}) \\
& \quad (\lambda z : \text{student}(z). \text{EVERY}_2^s[2, \text{THE}_2](\text{teacher})(\lambda y. \text{like}(z, y))) \\
& = \lambda x. \text{SOME}_1^s(\text{student})(\lambda z : \text{student}(z). \text{THE}_2(\text{teacher})(\lambda y. \text{like}(z, y))) \\
& = \lambda x : \text{teacher}(x). \text{SOME}_1^s(\text{student})(\lambda z : \text{student}(z). \text{like}(z, x))
\end{aligned}
\]

And in our final iteration of function application, we apply \(\mathllbracket \text{every}_2 \text{ teacher} \mathrrbracket^s\) to the predicate resulting from lambda abstraction:

(22) \[
\begin{aligned}
\mathllbracket \text{every}_2 \text{ teacher} \mathrrbracket^s(\mathllbracket \lambda_2 a_1 \text{ student } \lambda_1 a_1 \text{ student like every}_2 \text{ teacher} \mathrrbracket^s) & = 1 \text{ iff} \\
\text{EVERY}_2^s(\text{teacher})(\lambda x : \text{teacher}(x). \text{SOME}_1^s(\text{student})(\lambda z : \text{student}(z). \text{like}(z, x)))
\end{aligned}
\]

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 (22) uses \(\text{SOME}_1^s\) and \(\text{EVERY}_2^s\) instead of just \(\text{SOME}\) and \(\text{EVERY}\), the truth conditions of (22) 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 swap out \(\text{SOME}\) and \(\text{EVERY}\). We can easily define a swap state that does precisely this: namely, \text{stay} as defined in (23), which swaps out every etett for itself at every index:
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 (24):

\[
(24) \quad [[(1b)]^{\text{stay}} = 1 \text{ iff } \text{EVERY}(\lambda x: \text{teacher}(x). \text{SOME}(\lambda z: \text{student}(z). \text{like}(z, x)))]
\]

One may reasonably complain that requiring the state parameter to always be `stay` makes this parameter decidedly unparameterlike, as the apparent hallmark of a semantic parameter is that there is some flexibility in what that parameter can be set to. For this reason it is worth mentioning that I treat swap states as a parameter exclusively as a matter of formal and didactic convenience: it is an easy way to ignore swap states when they are irrelevant and use them when they are relevant. In reality, what is likely the best way to utilize swap states is to “reify” them in the semantics by including them as explicit arguments of (some) lexical items, then feeding `stay` to the resulting denotation at the end of the derivation. This reification has been executed in a variety of ways for variable assignments—see Charlow (2018) for a particularly elegant implementation—so given the parallels these methods will extend equally well to swap states. However, I will stick to parameterizing swap states as a matter of convenience.

It is worth noting that the interpretation in (24) can be further simplified. EVERY and SOME famously meet Keenan & Stavi’s (1986) condition of conservativity, defined in (25):

\[
(25) \quad D \text{ is conservative iff for all } A \text{ and } B, D(A)(B) \leftrightarrow D(A)(A \cap B)
\]

Put simply, every farmer smokes iff every farmer is a farmer who smokes, and a student of mine left iff a student of mine is a student of mine who left. Because of this, the domain restrictions imposed by THE in (24)—that is, the ones restricting `x` to teachers and `z` to students—are redundant, in that the conservativity of EVERY and SOME means that the same result would obtain if the restrictions were absent. Thus, (24) is equivalent to (26):

\[
(26) \quad [[(1b)]^{\text{stay}} = 1 \text{ iff } \text{EVERY}(\lambda x: \text{teacher}(x). \text{SOME}(\lambda z: \text{student}(z). \text{like}(z, x)))]
\]

More generally, following Keenan & Stavi (1986) conservativity is often stated as a universal holding of all determiner quantifiers, meaning that at least in terms of determiner quantification the domain restrictions imposed by THE make no direct semantic impact. In fact, Romoli (2015), building on work by Chierchia (1995), Fox (2002), and others, turns this all-too-convenient fact on its head, arguing that the copy theory of movement could be used to explain the conservativity generalization. In short, Romoli’s proposal is that if all quantificational DPs move at least once, then the restrictor of a lower copy will always be contained within the scope of the highest copy, and thus the scope of the quantifier always ends up being conjoined with the restrictor of that quantifier: `Det Restrictor Scope` is automatically interpreted as `Det Restrictor Restrictor+Scope`. As a result, any derivation using a non-conservative quantifier will end up equivalent to the same derivation using some (often
trivial) conservative quantifier. Not only is this an interesting proposal in its own right, but
the observation that copies can impose conservativity will become directly relevant when we
discuss comparatives in the next section, as the comparative morpheme -er has frequently
been given a non-conservative denotation.

Wrapping up, in this section I have proposed a theory of semantic composition in the
copy theory of movement that generates the semantic effects of trace conversion in a straight-
forwardly bottom-up compositional semantics, without either a syntactic or a semantic op-
eration of trace conversion. In the next section I will push this theory further by accounting
for modal and degree phrase scope-taking, which as discussed in the previous section pose
problems for (syntactic) trace conversion due to their lack of a convertible determiner.

3 Generalizing to modals and degree phrases

In this section I will show how the analysis developed in the previous section can be extended
to account for scope-taking by modals and degree phrases. We will see that once the analysis
is made appropriately type-polymorphic, modals and degree phrases can be captured in
essentially the same way as quantificational DPs.

3.1 Modals

I will use the sentences in (6), repeated below, to illustrate the analysis of modal scope:

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

I will use the following LFs for these sentences:

(27)  a. not [can1 res] mod Rivka leave the party
    b. [must1 res] λ_w,1 not [must1 res] mod Rivka leave the party

The morpheme res serves as the covert, contextually-determined restrictor for the modal's
world-quantification. In other words, it fills the role that on traditional Kratzerian accounts
of modals and conditionals is also played by if clauses (Kratzer 1981, 1991a,b, 2012). mod,
meanwhile, lambda abstracts over the evaluation world parameter, so that while Rivka leave
the party denotes a truth value for a given world of evaluation, mod Rivka leave the party
denotes a function from worlds to truth values. Notice also that in addition to its index, the
lambda abstractor λ_w,1 has a parameter for the type w of possible worlds, since we will now
be able to lambda-abstract over objects of various types. Thus, λ_1 and λ_2 in the previous
section will have to be replaced with λ_e,1 and λ_e,2.

7 This is a somewhat outdated view of how possible worlds/situations enter the compositional semantics,
with the more common view nowadays being that they are syntactically represented as pronouns that are
bound by lambda operators (see, e.g., Percus 2000). I leave for future work the issue of how such an approach
might be integrated with the analysis in this paper.
Our denotations for constituents will now be relative to two additional parameters: a context c and a world of evaluation u. The denotation of Rivka leave the party—that is, the part of the sentence below Mod—is a truth value, true iff Rivka leaves the party in u:

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

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

(29) \[ [\text{Mod } X]^{c,u,s} = \lambda v. [X]^{c,v,s} \]

(30) \[ [\text{Mod } \text{Rivka leave the party}]^{c,u,s} = \lambda v. \text{Rivka leaves the party in } v \]

Next up are can\textsubscript{1} and its restrictor res. On a traditional Kratzerian account, a conditional like (31a) receives the (approximate) interpretation in (31b), paraphrasable as (31c):

(31) a. If Mario comes, then Rivka can leave the party.

b. \[ [[[31a]]]^{c,u} = 1 \text{ iff } \exists v \in \text{Best}^{c}_{u}(\text{Dom}^{c}_{u} \cap \text{mcome})[\text{rleave}(v)], \]

where Dom\textsubscript{u} is the set of worlds circumstantially accessible from u, and Best\textsubscript{c}(A) is the set of ideal worlds in A given the standards in u.

c. Among those worlds that are (i) circumstantially accessible, (ii) worlds in which Mario comes to the party, and (iii) such that the rules are maximally obeyed given (i) and (ii), there is at least one world in which Rivka leaves the party.

If we say that the conditional antecedent's restriction is always present—whether overtly or as the silent res—then on a traditional account [can]\textsuperscript{c,u} is a relation between propositions:

(32) \[ [\text{can}]^{c,u}_{\text{take } 1} = \lambda w_{t} \lambda q_{w_{t}}. \exists v \in \text{Best}_{u}^{c}(\text{Dom}_{u}^{c} \cap p)[q(v)] \]

So how do we integrate this into the present analysis, given that modals can take scope via movement? On the definition in (32), [can] is a (wt)(wt)t-type existential quantifier, albeit with some extra baggage in the form of Best and Dom, while our definition of [a] is an (et)(et)t-type existential quantifier. Now let us suppose that SOME and EVERY, rather than being strictly (et)(et)t-type quantifiers, are instead type-polymorphic (at)(at)t quantifiers (hereafter TPQs):

(33) a. SOME := \lambda f_{t} \lambda J_{t}\subseteq J. \neg (J \cap J' \neq \emptyset)

b. EVERY := \lambda f_{t} \lambda J_{t}. J \subseteq J'

Suppose in addition that swap states only trade in TPQs: they take an index and a TPQ and return a (possibly identical) TPQ. Notice that none of this means that [a] is itself a TPQ, in spite of the fact that it uses TPQ SOME, since the definition in (12a) lexically restricts [a] to taking et-type arguments. Our [can] will also utilize the TPQ SOME, but will lexically restrict its arguments to being type wt. More specifically, we define [can] as in (34):

(34) \[ [\text{can}^{n}_{n}]^{c,u,s} = \lambda p_{w_{t}} \lambda q_{w_{t}}. \text{SOME}^{c}_{n}(\text{Best}^{c}_{u}(\text{Dom}_{u}^{c} \cap p))(q) \]

We now have enough to continue with our derivation. [res]\textsuperscript{c,u,s} returns a contextually determined restrictor proposition R\textsuperscript{c}, so the result of composing can with res is as in (35):
For this conundrum can be resolved as follows: deal in TPQs, THE to be parameterized to objects of arbitrary type. However, because swap states now only individual. Since we can now lambda abstract over objects of arbitrary type, THE must be able that in the definition of THE in the previous section, it is a etett parameterized to an indet. We thus start by defining a TPQ version of THE, which presents a bit of a problem. Recall The next step is to lambda abstract, but while swap states and the TPQs EVERY and SOME that in the definition of THE in the previous section, it is a etett parameterized to an indet. We thus start by defining a TPQ version of THE, which presents a bit of a problem. Recall the definition in (41), THEₖ is genuinely type-polymorphic, and can thus be the output of a swap state: if k is type β and J and J' are not type βt but, say, type δt, there is not a type mismatch when composing with THEₖ. However, the output is nonetheless undefined. BD and BD′ in (15) can be similarly redefined.
With our new THE in tow, redefining lambda abstraction becomes a simple task, with
the only real difference being that the resulting type of lambda abstraction depends on the
type parameter of the \(\lambda\) node. This is shown in (42):

(42) **New Lambda Abstraction (Revised):**
\[
\lambda_{\beta,n} X^{C,s} = \lambda_{k} [X]^{C,s[n,\text{THE}]} \\
\]

We can now continue with the derivation of (6b):

(43) \[
\lambda_{w,1} \text{not must}_1 \text{RES MOD Rivka leave the party}^{C,s} \\
= \lambda v \text{.} \text{not must}_1 \text{RES MOD Rivka leave the party}^{C,s[1,\text{THE}]} \\
= \lambda v \text{.} \text{¬EVERY}^{s[1,\text{THE}]}(\text{BEST}^{C}_{u}(\text{DOM}_{u}^{c} \cap R^{c}))(r\text{leave}) \\
= \lambda v \text{.} \text{¬THE}_{u}(\text{BEST}^{C}_{u}(\text{DOM}_{u}^{c} \cap R^{c}))(r\text{leave}) \\
= \lambda v : v \in \text{BEST}^{C}_{u}(\text{DOM}_{u}^{c} \cap R^{c}) \cdot \text{¬rleave}(v)
\]

This gives us a proposition that is domain-restricted to ideal worlds, and is true of those
worlds in which Rivka does not leave. This can then be fed back into \([\text{must}_1 \text{RES}]^{C,s}\):

(44) \[
\text{must}_1 \text{RES}^{C,s}(\lambda_{w,1} \text{not must}_1 \text{RES MOD Rivka leave the party})^{C,s[1,\text{THE}]} = 1 \iff \\
\text{EVERY}^{s}(\text{BEST}^{C}_{u}(\text{DOM}_{u}^{c} \cap R^{c}))(\lambda v : v \in \text{BEST}^{C}_{u}(\text{DOM}_{u}^{c} \cap R^{c}) \cdot \text{¬rleave}(v))
\]

And as always we evaluate with the state parameter \(\text{stay}\), getting us our final denotation in
(45a), which thanks to the conservativity of EVERY is equivalent to (45b):

(45) \[
(6b)^{C,s,\text{stay}} = 1 \iff \\
a. \text{EVERY}(\text{BEST}^{C}_{u}(\text{DOM}_{u}^{c} \cap R^{c}))(\lambda v : v \in \text{BEST}^{C}_{u}(\text{DOM}_{u}^{c} \cap R^{c}) \cdot \text{¬rleave}(v)) \\
b. \text{EVERY}(\text{BEST}^{C}_{u}(\text{DOM}_{u}^{c} \cap R^{c}))(\lambda v \cdot \text{¬rleave}(v))
\]

The interpretation is the desired one: in all ideal worlds, Rivka does not leave, i.e., Rivka is
required not to leave.

In summary, by adopting an approach to swap states in which what they swap in and out
are type-polymorphic quantifiers, the same principles that accounted for the scope of DP
quantifiers in the previous section can account equally well for the scope of modals. Next
we turn to degree phrases.

### 3.2 Degree phrases

#### 3.2.1 The problem of non-conservative er

Before offering an analysis of degree phrases, it will help to discuss a somewhat more tradi-
tional, copy-free analysis of comparatives. We will use (46) as our sample sentence:

(46) Jo is taller than Al is.

On a traditional account, (46) has an LF along the lines of (47):

(47) \[e_{r1} \text{ Op}_{2} \lambda_{2} \text{ than Al is} \; t_{2} \] \[\lambda_{1} \text{ Jo is tall} \; t_{1}\]
[tall] is a relation between a degree \( d \) and individual \( x \), true iff \( x \) is at least \( d \)-tall:

\[
(48) \quad \text{[tall]}_{\text{traditional}} = \lambda d \lambda x. \text{height}(x) \geq d
\]

\( Op \) is a \( wh \)-like operator that triggers lambda abstraction, while \( than \) is generally treated as semantically vacuous. Thus, the denotation of the restrictor of \( er \) is a degree predicate true of \( d \) iff \( Al \) is at least \( d \)-tall, and the denotation of the scope of \( er \) is a degree predicate true of \( d \) iff \( Jo \) is at least \( d \)-tall. \( [er] \) is thus of type \((dt)(dt)t\), i.e., a degree quantifier.

Two common denotations for \( er \) that generate the correct truth conditions for (46) are provided in (49).

\[
(49) \quad \begin{align*}
\text{a. } [er]_{\text{take 1}} & = \lambda D \lambda D'. \max(D) < \max(D') \\
\text{b. } [er]_{\text{take 2}} & = \lambda D \lambda D'. \exists d[\neg D(d) \land D'(d)]
\end{align*}
\]

The definition in (49a) states 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. Meanwhile, (49b) states that 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.

A noteworthy fact about both definitions in (49) is that they are not conservative. For instance, if (49a) were conservative, we would expect it to be equivalent to (50):

\[
(50) \quad \lambda D \lambda D'. \max(D) < \max(D \cap D')
\]

But since \( D \) is necessarily a superset of \( D \cap D' \), it is impossible for the maximal degree in the latter to exceed the maximal degree in the former. Thus, not only is (50) not equivalent to (49a), but it is contradictory. Similarly, if (49b) were conservative, we would expect it to be equivalent to (51a), which is in turn equivalent to (51b):

\[
(51) \quad \begin{align*}
\text{a. } & \lambda D \lambda D'. \exists d[\neg D(d) \land [D \cap D'](d)] \\
\text{b. } & \lambda D \lambda D'. \exists d[\neg D(d) \land D(d) \land D'(d)]
\end{align*}
\]

Thus, contradiction again arises.

If degree phrases take scope by means of QR, then this non-conservativity presents a \textit{prima facie} problem. As discussed above, if the restrictors of lower copies make semantic contributions, conservativity is essentially imposed: thanks to THE, the degree predicate in the scope of the degree phrase will be restricted to degrees not exceeding \( Al \)'s height, thereby generating one of the contradictory interpretations in (50) or (51).

To further illustrate, say we convert the degree quantifier in (49b) into a TPQ; following Romoli (2015) I call this SOMENON:

\[
(52) \quad \text{SOMENON} := \lambda J_{al} \lambda J'_{al}. \exists k_{al}[\neg J(k) \land J'(k)]
\]

The interpretation we derive by composing the structure in (47) is the perfectly reasonable (53), true iff there is a degree \( d \) of height such that \( Al \) does not reach \( d \) and \( Jo \) does.

\[
(53) \quad \text{SOMENON}(\lambda d. \text{height(al)} \geq d)(\lambda d. \text{height(jo)} \geq d)
\]
However, if there is a lower copy of the restrictor of \(er\), then THE in combination with this lower copy will restrict the domain of the scope of \([er]\):

\[(54)\quad \text{SOMENON}(\lambda d. \text{height}(al) \geq d)(\lambda d : \text{height}(al) \geq d. \text{height}(jo) \geq d)\]

This generates the contradictory reading seen in \((51b)\).

So how do we avoid this? Two possibilities immediately present themselves: either the restrictor is for some reason not actually interpreted in the lower copy, or the degree quantifier itself is actually conservative, with some other factor giving the illusion of a non-conservative degree quantifier. In the rest of this section I will utilize both of these techniques, incorporating syntactic insights from Bhatt & Pancheva (2004) and a semantic analysis due to Schwarzschild (2008) and Gajewski (2008); each technique suffices on its own, but each is also motivated on independent grounds. While this will not lead to a definitive theory of the syntax and semantics of comparatives—for instance, it excludes differential comparatives like \textit{Jo is two inches taller than Al is}—it does illustrate that comparatives may not be as problematic as they first appear.

\subsection{Decomposing \(er\)}

Suppose that we think of SOMENON not as a single quantificational element, but rather as a combination of two semantic components: the conservative SOME and a negation in the restrictor. Now suppose further that this is actually reflected in the LF of \((46)\), meaning that the traditional LF in \((47)\) is replaced with the slightly less traditional LF in \((55)\):

\[(55)\quad [\text{cmp}_1 \text{op}_2 \lambda_2 \text{than } \text{neg} \text{ Al is tall } t_2] \lambda_1 \text{ Jo is tall } t_1\]

If \([\text{neg}]\) is boolean negation and \([\text{cmp}]\) is simply SOME (restricted to degree quantification), we generate the same interpretation as we did using the LF in \((47)\) and the denotation for \(er\) in \((49b)\), except by different means: this time, the degree quantifier is SOME, and its restrictor is the set of degrees \textit{not} less than or equal to Al's height.

This small change makes all the difference with respect to the problem stated above. To see why, consider \((56)\), the interpretation we get with the LF in \((55)\):

\[(56)\quad \text{SOME}(\lambda d. \text{height}(al) \not\geq d)(\lambda d : \text{height}(al) \not\geq d. \text{height}(jo) \geq d)\]

The truth conditions here are identical to those in \((53)\). But this time, if a lower copy of the restrictor is interpreted in the scope of the degree quantifier, we get \((57)\) instead of \((54)\):

\[(57)\quad \text{SOME}(\lambda d. \text{height}(al) \not\geq d)(\lambda d : \text{height}(al) \not\geq d. \text{height}(jo) \geq d)\]

Not only is \((57)\) not contradictory, in contrast to \((54)\), but thanks to the conservativity of SOME it is equivalent to \((56)\). Thus, by decomposing \(er\) into two heads \text{cmp} and \text{neg}, we generate the same semantic result, but without the potential problem of copy-induced contradiction faced by SOMENON.

This is all well and good, but is there independent evidence for decomposing \(er\) in this manner? Thankfully, yes: as first discussed in detail by Larson (1988) and further elaborated upon by Schwarzschild & Wilkinson (2002) (and others since), more traditional analyses
face difficulties when interpreting quantificational elements in comparison clauses, as in (58a), which on traditional accounts would have the LF in (58b):

(58)  a. Jo is taller than every student is.
       b. [er₁ Op₂ λ₂ than every student is tall]₁ Jo is tall ₁

Depending on if we choose (49a) or (49b) for [er], we predict to get the (semi-formal) interpretation in (59a) or (59b), respectively.

(59)  a. max(λd. EVERY(student)(d-tall)) < max(λd. Jo is d-tall)
       b. ∃d[¬EVERY(student)(d-tall) ∧ Jo is d-tall]

But these generate the wrong truth conditions. Intuitively, (58a) is true iff for each student, Jo is taller than that student. In other words, Jo is taller than the tallest student. But notice that in (59a), the degree that Jo’s height is compared to is the maximal d such that every student reaches d, meaning we are comparing her not to the tallest student, but to the shortest student. In other words, we wrongly predict (58a) to mean that Jo is taller than at least one student. The same issue arises in (59b), which is true iff there is a degree of height that Jo reaches and that at least one student fails to reach.

Instead, in order to generate the desired reading of (58a) with an LF like (58b), every student must QR out of the comparison clause:

(60)  [every₃ student] λ₃ [er₁ Op₂ λ₂ than t₃ is tall]₂ λ₁ Jo is tall ₁

But as noted by Schwarzschild & Wilkinson (2002), this is syntactically implausible, as comparison clauses are movement islands:

(61)  * Who₁ is Jo taller than t₁ is?

We thus appear to be at an impasse: quantifiers in degree phrases must outscope those degree phrases, but cannot undergo the operation required to do so. However, Schwarzschild (2008) and Gajewski (2008) observe that if we look at the representation in (59b), every student does not actually need to outscope the degree quantification in order to get the correct interpretation: it only needs to outscope negation. In other words, the semi-formal (62) does get the right truth conditions:

(62)  ∃d[EVERY(student)(not d-tall) ∧ Jo is d-tall]

By syntactically decomposing er into CMP and NEG, such an interpretation can be generated through scope-splitting: every student outscopes NEG while staying inside the restrictor of CMP.

(63)  [CMP₁ Op₂ λ₂ than [every₃ student] λ₃ NEG t₃ is tall]₂ λ₁ Jo is tall ₁

The Schwarzschild-Gajewski analysis of comparatives is not the only scope-splitting analysis on the market, and there is a sense in which it is particularly convenient: it is both maximally simple and wholly faithful to an extended conservativity hypothesis in which all (αt)(at)ₜ-type quantifiers—or at least all of those that undergo QR—are conservative. Not all scope-splitting analyses stick to conservative degree quantifiers: Heim (2006b), for example, utilizes an operator with the non-conservative denotation in (64).
My goal in this paper is not to hash out the differences between various scope-splitting analyses of comparatives. Rather, the point is to emphasize that once one accepts split er, as is independently motivated to account for the interpretation of quantificational elements in comparison clauses, it is no longer obvious that a proper account of comparatives requires the use of a non-conservative degree quantifier. With this in mind, I will stick to the Schwarzschild-Gajewski analysis.

3.2.3  Late merging the than clause

Before we build and test our semantics for comparatives, we need a syntactic representation. We know that for our sentence in (46), we want an LF that on a traditional copy-free approach would look like (55), repeated below:

I will adopt a syntax more or less along lines proposed by Bhatt & Pancheva (2004), with a couple of minor changes. One of the guiding observations made by Bhatt & Pancheva (2004) is that while than clauses are interpreted as the complement of er, the former (almost) always have to undergo extraposition, i.e., the same syntactic process as that separating the relative clause from book in (65).

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

In cases like (46) this extraposition is seemingly string-vacuous, but in nominal comparatives this is no longer the case.

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

If we adopt the common view that more is much/many + er, then given that the than clause is the sister of er the ill-formed (66a) should be the result of a derivation without than clause extraposition. Once the than clause is extraposed we get the well-formed (66b).

Bhatt & Pancheva (2004) provide robust evidence that than clauses are indeed extraposed, based on a variety of parallels between than clauses and extraposed relative clauses of the sort seen in (65). They then adopt the analysis of extraposition proposed by Fox & Nissenbaum (1999) and Fox (2002), in which QR has a pivotal role to play. In short, Fox & Nissenbaum follow Lebeaux (1990) in positing that adjoined constituents like relative clauses can be late merged, i.e., countercyclically merged to the constituent they modify after that constituent has already merged into a larger structure and undergone movement. Extraposition is then the result of late-merging the relative clause after QR has taken place; while only the lower copy of the determiner and its nominal complement is pronounced, the relative clause is merged with the higher (unpronounced) copy of the NP and pronounced there. If QR is stipulated to be a rightward operation, this generates the correct word order:

(67)  Rivka read [every book] yesterday [every book that Kwame had suggested to her]
Among other motivations, this accounts for what they call Williams’s Generalization (after Williams (1974)), which states that a DP must scope at least as high as anything extraposed from it. An illustration of Williams’s Generalization can be seen in (68):

(68) Illustration of Williams’s Generalization (Fox 2002, p. 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.

(68a) is ambiguous between two readings. If every scopes below before, the resulting interpretation is that I was the first to make my way through John’s whole list, though certain books you might have finished before me. If every scopes above before, the interpretation is that each book in the list was finished by me first. But (68b), in which the relative clause is extraposed past before, is unambiguous, and only the latter reading is available.

With this in mind, Bhatt & Pancheva (2004) propose that in comparatives the than clause is late merged with er, so that only er is merged in the lower position.8 In (69) this is roughly illustrated for the nominal comparative in (66b):

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

So how are such structures interpreted? Let us take (46) as our example. In this case, the derived LF will look like (70a). (I return to displaying QR as leftward for ease of reading.) The struckthrough tall er1 in (70a) indicates resolution of the ellipsis in than Al is. After inserting lambda abstraction nodes and performing trace conversion, the final interpreted LF is as in (70b).

(70) a. [er1 than Al is tall er1] Jo is tall er1
   b. [er1 λd1 than Al is tall the d1] λd1 Jo is tall the d1

Assuming one or another version of non-conservative [er], the LF in (70b) generates the same correct interpretation as the more traditional LF in (47), repeated below.

(47) [er1 Op2 λ2 than Al is tall t2] λ1 Jo is tall t1

Now we revise Bhatt & Pancheva’s (2004) analysis in order to better suit our compositional needs. The first change is that as per the above discussion, er is broken up into the degree-quantificational cmp and negation neg. The second change is that we of course no longer perform trace conversion, and all copies of cmp are interpreted. To see what the third change is, notice that in the pre-trace-conversion LF in (70a), the lower copies of er have no complement, meaning that they have no restrictor. If we were to import this directly into our analysis, it would mean that cmp has no restrictor at its lower copies, as in (71).

(71) [cmp1 λd,1 than neg Al is tall cmp1] λd,1 Jo is tall cmp1

For Bhatt & Pancheva it is okay for er to not have a restrictor, since trace conversion will render the result compositionally tractable (though at the cost of the syntactic stipulation

---

8Note that this means that complements can also be late merged, in contrast to Lebeaux’s (1990) assertion that it is only adjuncts that can be late merged.
that *er can be replaced with a definite determiner). But for us this is not an option: the composition breaks down if any copy of \( \text{cmp} \) does not have its first argument saturated.

There are a few ways this could be resolved. The first is to stipulate that a lexical item \( \text{vac} \), whose denotation is a vacuously true degree predicate, is inserted as the complement to lower copies of \( \text{cmp} \).

(72) \( [\text{vac}]^s = \lambda d. \top \)

However, it is unclear how such lexical insertion could be performed in a way that does not do injustice to contemporary syntactic assumptions. A second possibility is to add a semantic rule that performs the equivalent of \( \text{vac} \)-insertion in cases where a phrase-level constituent has a restrictor-less denotation:

(73) **RESTRICTION INSERTION (RI):**
    
    If \( [\text{XP}]^s \) is of type \( (\beta l)(\beta l)t \), then \( [\text{XP} \ Y]^s \) is the result of composing \( [Y]^s \) with \( [\text{XP}]^s(\lambda k. \top) \).

A third possibility is that \( \text{vac} \) (or something like it) is actually present in both the higher and lower copies; in the lower copy it restricts \( \text{cmp} \) on its own, and in the higher copy \( \text{vac} \) and the \( \text{than} \) clause compose via Predicate Modification, but since \( \text{vac} \) is true of all degrees the result of intersection is the same as the interpretation of the \( \text{than} \) clause on its own. This seems to be the least stipulative approach, as it appears syntactically plausible and does not require the addition of a new compositional rule. Nonetheless, in the rest of this paper I will adopt the first analysis (\( \text{vac} \)-insertion), solely for the sake of simplicity and brevity. Thus, the LF we will be interpreting is the one in (74):\(^9\)

(74) \[ \text{cmp}_1 \lambda d,1 \text{ than neg Al is tall [cmp}_1 \text{ vac}] \lambda d,1 \text{ Jo is tall [cmp}_1 \text{ vac} \]

### 3.2.4 Composition

Now we are left with the task of composing the interpretation for the LF in (74). We start by composing the \( \text{than} \) clause. As mentioned before, a traditional analysis would assign \( \text{tall} \) the denotation in (48), true of a degree \( d \) and individual \( x \) iff \( x \) is at least \( d \)-tall.

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

However, in much the same way that the verb *like* had to be type-lifted to allow composition with quantificational DP arguments, now *tall* has to be type-lifted to allow composition with quantificational DegP arguments. This lifted version can be seen in (75). (Note that from here on I exclude the context and world of evaluation parameters as they are irrelevant, keeping only the swap state parameter.)

---

\(^9\)One may wonder how lambda abstraction is triggered inside the degree phrase in (74), since no operator undergoes movement. However, something like the analysis of relative clauses adopted by Sauerland (1998, 2004) and Fox (2002) could work: the elided \( \text{cmp}, \text{vac}, \) or phrase containing both undergoes \( \text{wh} \)-movement and triggers lambda abstraction, but is not interpreted at the post-movement site:

(i) \[ \text{cmp}_1 \text{ vac} \lambda d,1 \text{ than neg Al is tall [cmp}_1 \text{ vac}] \lambda d,1 \text{ Jo is tall [cmp}_1 \text{ vac} \]

This would explain the lambda abstraction in addition to the well-known island sensitivity of comparatives.
As mentioned above, cmp contributes existential degree-quantification:

(76) \[ \begin{align*}
[\text{cmp}]^s & = \lambda d \lambda d'. \text{SOME}_n^s(D)(D')
\end{align*} \]

First \[ [\text{cmp}] \] takes \[ [\text{vac}] \] as an argument, and then \[ [\text{tall}] \] takes the output as an argument. The result is as in (77):

(77) \[ \begin{align*}
[\text{tall cmp}_1 \text{ vac}]^s & = \lambda x. \text{SOME}_n^s(\lambda d. \top)(\lambda d. \text{height}(x) \geq d)
\end{align*} \]

We next combine with \[ [\text{Al}] \] which I take to simply be the individual al. (Things get more complicated, though still manageable, if we assume that names can take scope via movement, and thus need to have state-sensitive interpretations.) This gets us (78):

(78) \[ \begin{align*}
[\text{tall cmp}_1 \text{ vac}]^s(\text{[Al]})^s & = 1 \text{ iff } \text{SOME}_1^s(\lambda d. \top)(\lambda d. \text{height}(\text{al}) \geq d)
\end{align*} \]

This then combines with neg, which contributes boolean negation:

(79) \[ \begin{align*}
[\text{neg}]^s(\text{[Al is tall cmp}_1 \text{ vac}]^s) & = 1 \text{ iff } \neg \text{SOME}_1^s(\lambda d. \top)(\lambda d. \text{height}(\text{al}) \geq d)
\end{align*} \]

Sticking with our assumption that than is semantically vacuous, the next step is lambda abstraction, which completes the derivation of the restrictor of the highest copy of cmp$_1$:

(80) \[ \begin{align*}
[\lambda _{\text{d,1 than neg Al is tall cmp}_1 \text{ vac}]^s
& = \lambda d. [\text{than neg Al is tall cmp}_1 \text{ vac}]^s[1,\text{THE}_d]
& = \lambda d. \neg \text{SOME}_1^s[1,\text{THE}_d](\lambda d'. \top)(\lambda d'. \text{height}(\text{al}) \geq d')
& = \lambda d. \neg \text{THE}_d(\lambda d'. \top)(\lambda d'. \text{height}(\text{al}) \geq d')
& = \lambda d : \top. \neg (\text{height}(\text{al}) \geq d)
& = \lambda d. \text{height}(\text{al}) < d
\end{align*} \]

The derivation of the scope of the degree phrase is identical, except replacing Al with Jo and excluding negation. The result is in (81):

(81) \[ \begin{align*}
[\lambda _{\text{d,1 Jo is tall cmp}_1 \text{ vac}]^s
& = \lambda d. \text{height}(\text{jo}) \geq d
\end{align*} \]

As our final step, the restrictor and scope of the highest copy of cmp$_1$ compose with the quantificational element, deriving the final interpretation in (82):

(82) \[ \begin{align*}
[74]^s & = 1 \text{ iff } \text{SOME}_1^s(\lambda d. \text{height}(\text{al}) < d)(\lambda d. \text{height}(\text{jo}) \geq d)
\end{align*} \]

Since we always evaluate with respect to the state stay, we generate the following truth conditions:

(83) \[ \begin{align*}
[74]^{\text{stay}} & = 1 \text{ iff } \text{SOME}(\lambda d. \text{height}(\text{al}) < d)(\lambda d. \text{height}(\text{jo}) \geq d)
\end{align*} \]

This is the desired result: we predict truth iff there is a degree of height that exceeds Al’s height, but does not exceed Jo’s.

We thus see that by adopting a sufficiently nuanced view of the syntax and semantics of comparatives, scope-taking of degree phrases can be readily accounted for in the present
theory, in spite of our initial concerns based on the apparent non-conservativity of \([\text{er}]\). In fact, there were two separate and independently motivated reasons not to be afraid of \(\text{er}\): it is actually composed of two parts, including a degree quantifier that may actually be conservative, and the restrictor is not interpreted in the lower copy.\(^{10}\)

### 4 Concluding remarks

In this paper I have provided a semantics for copy composition in which quantificational elements provide their own quantificational force (contra Johnson) and lower copies receive bound variable interpretations strictly by means of bottom-up compositional principles (contra Fox). This analysis was shown to extend beyond the narrow purview of quantificational DPs, also being able to account for scope-taking movement by modals and degree phrases. In tying a bow on this paper, I will discuss a couple of areas that seem to me to be worth exploring in future work.

The first and perhaps most obvious issue is empirical coverage. While I have attempted to illustrate the broad applicability of the analysis in this paper by extending its scope beyond quantificational DPs, there nonetheless remain gaps that need to be filled, such as wh-phrases, adverbs of quantification (e.g., always, usually), and operators that quantify over focus alternatives (only, even). In addition, the analysis in this paper must be integrated with an appropriate theory of pronominal binding; if one adopts a dynamic approach (see Kamp 1981, Heim 1982, and many since), this might entail replacing our static TPQs with dynamic TPQs. These topics are left for future exploration.

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

\[
\text{(84) Abstract and Apply (AA):} \\
\text{If } [X_n]^s \text{ is type } (\alpha \beta) \gamma, \text{ and } [Y]^s \text{ is type } \beta, \text{ then} \\
[X_n Y]^s = [X_n]^s(\lambda k_{\alpha}. [Y]^s[n, \text{THE}_k])
\]

This rule seems to generate the right results for all of the cases at hand without the use of lambda-abstracting nodes, but questions may arise about its possible stipulativeness. While

\(^{10}\)It is worth noting that the alleged non-conservativeness of \([\text{er}]\) is crucial to one aspect of Bhatt & Pancheva’s (2004) theory. They argue that degree phrases must scope exactly as high—not just at least as high, as per Williams’s generalization for DP quantifiers—as the extraposition site of the than clause. They account for this by exploiting the fact that if \([\text{er}]\) is non-conservative, then moving the degree phrase after the than clause has been merged generates precisely the contradictory reading we have avoided. My analysis does not make this prediction, as nothing goes wrong if the degree phrase undergoes further movement. If Bhatt & Pancheva’s empirical observation is correct, then either something else must freeze the degree phrase’s scope, or CMP and NEG should be replaced with an alternative that retains a non-conservative degree quantifier while still allowing split scope. I leave a resolution of this matter for future work.
I do not find this rule particularly odious, perhaps a more palatable alternative can be found that does not make recourse to a new rule of semantic composition. But again, this must be left for another time.

References


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


Heim, Irene. 2006b. Remarks on comparative clauses as generalized quantifiers. MIT, Ms.


