564 Lecture 25 Nov. 23, 1999

1 Continuing note on presuppositional vs. nonpresuppositional dets.

Here's the argument about the nonpresupp vs. presupp analysis of "every" that I couldn't reconstruct last time.

1. (a) If the presupp. analysis of determiners is right, then surely we should judge sentences like (i) below, where the restrictor has no extension, as neither true nor false, but rather undefined:

i. Every unicorn has one horn.

(b) The fact that we generally judge a sentence like (i) "true" seems to indicate that there is no presupposition of existence when we use "every", and the Aristotelian inference patterns are in fact conditional, rather than universal.

(c) But what about sentences like (ii) or (iii)?

ii. Every unicorn has two horns.

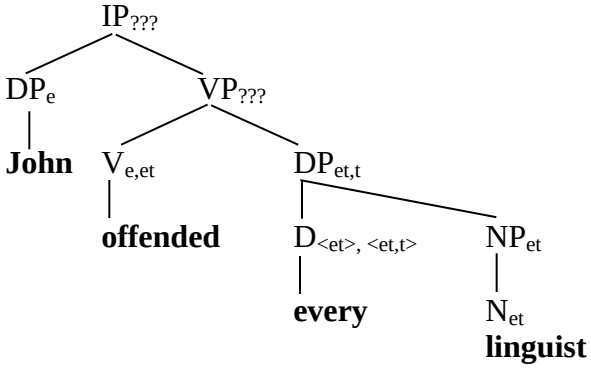
iii. No unicorn has one horn.

(d) If the "true" judgment on (i)is an argument against the treatment where there is presupposition of existence for the restrictor, then shouldn't the "false" judgement in (iii) argue against this treatment? For if there are no unicorns, then any statement made about the class of unicorns is predicted to be true by the non-presuppositional theory (This is why the Aristotelian conclusions don't follow in the non-presuppositional theory: when there are no instances of X, the non-presupp theory predicts sentences like "Every X is Y" are always true). So the fact that speakers judge ii. false demonstrates that something else is going on here; it's not that they're using the non-presuppositional versions of determiners, but rather that they know what unicorns, if there were any, would be like.

2 Object quantifiers?

So far, our quantifiers are of type <<e,t>,<<e,t>,t>>>. (For convenience's sake, I'll write this as <et, <et,t>> from now on), and they express a relationship between the restrictor set and the set denoted by the VP. What happens if we put a quantifier in object position? Bad, bad things.

2. (a) John offfended every linguist.



When we come to interpret the VP node, it has two daughters, neither of which can take the other as its argument. The VP node, then, will fail to get an interpretation. How can we avoid this bad consequence?

3 Potential answer #1: Type shifting quantifiers

Recall our first attempt at resolving the problem with gradable adjectives like "big" was to type-shift them, which enabled us to make reference to the function with which they were combining in the truth-conditions we supplied for the truth of the adjective. This approach entailed that every adjective had two entries, a lower type for sentences like 'John is big' and a higher type for 'Felix is a big cat'. A potential response to the quantifier problem here is similar: what if we proposed that quantifiers had two (or actually, several) different types, depending on the position in the argument structure that they occupy. In our example in (2) above, we certainly want the VP to be of type <et>, so we should propose a type-shifted version of the quantifier **every** of type <et, <<e,et>, et>>:

3. (a) [[**every**]] < et, <<e,et>, et>>

[λf D_{et} . [λg $D_{<e,et>}$ **.** [λx D_e **.** for every y {y: f(y) = 1}, g(y)(x)=1]]]

(b) Let's consider how this works for (2) above, where we have the following lexical entries:

[[**John**]] = John
[[**offended**]] = [λx D_e . [λy D_e **.** y offended x]]
[[**linguist**]] = [λx D_e . x is a linguist]

[[IP]] =
=[[VP]]([[**John**]]) by FA (on IP) and NN
=[[DP]]([[**offended**]])([[**John**]]) by FA (on VP) and NN
=[[**every**]]([[**linguist**]])([[**offended**]])([[**John**]]) by FA (on DP) and NN
=[λf D_{et} . [λg $D_{<e,et>}$ **.** [λx D_e **.** for every y {y: f(y) = 1}, g(y)(x)=1]]]
([[**linguist**]])([[**offended**]])([[**John**]]) by LT on [[**every**]]
= [λg $D_{<e,et>}$ **.** [λx D_e **.** for every y {y: [[**linguist**]](y) = 1}, g(y)(x)=1]]
([[**offended**]])([[**John**]]) by def. of λ on λf and [[**linguist**]]
= [λg $D_{<e,et>}$ **.** [λx D_e **.** for every y {y: [λz D_e . z is a linguist](y) = 1}, g(y)(x)=1]]
([[**offended**]])([[**John**]]) by LT on [[**linguist**]]
= [λg $D_{<e,et>}$ **.** [λx D_e **.** for every y {y: y is a linguist}, g(y)(x)=1]]
([[**offended**]])([[**John**]]) by def. of λ on λz
= [λx D_e **.** for every y {y: y is a linguist}, [[**offended**]](y)(x)=1]]([[**John**]])
by def of λ on λg and [[**offended**]]
= [λx D_e **.** for every y {y: y is a linguist}, [λw D_e . [λz D_e **.** z offended w]]
(y)(x)=1]]([[**John**]]) by LT on [[**offended**]]
= [λx D_e **.** for every y {y: y is a linguist}, x offended y]([[**John**]])
by def λ on λw and λz
= 1 iff for every y {y: y is a linguist}, [[**John**]] offended y

by def. λ on λx
= 1 iff for every y {y: y is a linguist}, John offended y
by LT on [[**John**]]
i.e. [[**John offended every linguist**]] = 1 iff John offended every linguist.

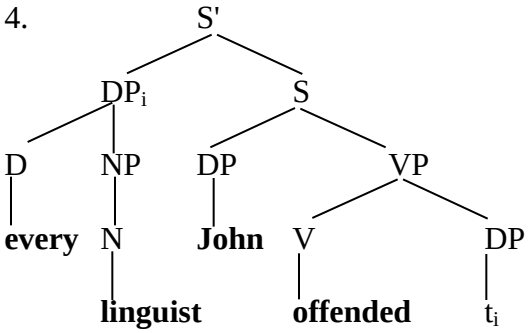
H&K note that it's not so crazy to assume there can be lots of types for a particular class of lexical items; recall our discussion of the types of connectives like **and** and **or**. (Interesting side note: what are possible types for **but**?). So this is a possible approach. (Cf, say H&K, Combinatorial Logic, and the work of Steedman, Jacobson, Szabolcsi and Cresswell).

4 Potential Answer # 2: Quantifier Movement

Another possible approach is to assert, well, if this structure is not interpretable, then this is not the structure which is input to the interpretation function. Rather, some further structural operations occur to create a structure where the quantifier has the appropriate types of arguments to combine with. These structural operations occur after Spell-Out (in English, at least) and hence are not reflected in the word order etc.

Garden-variety quantifier movement usually assumes simple movement of the QP to SpecCP, or perhaps adjunction of the QP to CP. However, that's not quite going to be enough to create an interpretable structure. Consider the plain movement structure in (4):

4.

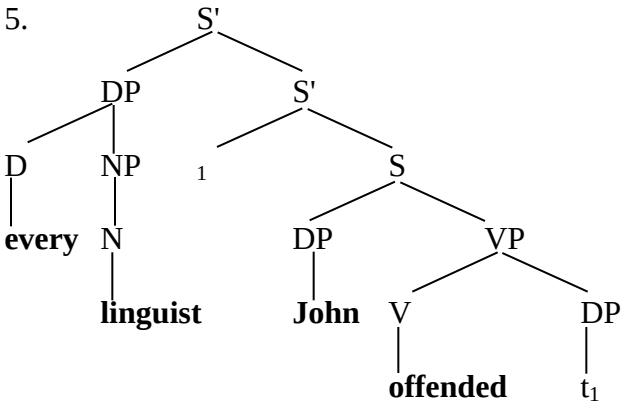


As it stands, is this structure interpretable by our semantic component? The answer at the moment is no. Consider what happens if we assume, as we have been doing, that traces are variables. The type of S will be <t>, because the verb has the appropriate number of arguments, hence giving a saturated expression. The DP **every linguist**, then, will try to combine with something of type t, which is not an appropriate argument for it, and we won't end up with a denotation.

However, we do have in our arsenal of operations something that can repair this structure. What we want is for the sister node to DP_i to have type <et>. What can we do to give it that type?

The answer is, we can do predicate abstraction over it. Let's assume that the structure which is created by movement is not 4, but rather something like 5:

5.



One way to think about this is that the index of movement is not *attached* to the DP which moves, but rather adjoined right below it. What will happen in this scenario is that we'll apply PA at S', which will give us a type <et> node, and then "every linguist" will be able to combine with the S' node in the usual way.

We'll have to write our P.A. rule in such a way that it's not just looking for a "relative pronoun" with an index, or "such" with an index, but rather, it's just looking for an index. Here's the new rule:

6. *Predicate Abstraction*

Let α be a branching node with daughters β and γ, where β dominates only a numerical index i. Then, for any variable assignment a, $[[\alpha]]^a = [\lambda x \quad D \text{ . } [[\gamma]]^{a^{x/i}}]$.

(Now, to interpret our relative clauses, we're going to have to assume that the relative pronouns and "such" are semantically vacuous, so that in fact all the interpretation function sees is the index – so it'll apply the above version of PA happily. If you want to see how that will work in the case of relative clauses, we'll come back to it later. There is a somewhat weird thing about it).

So, let's consider how this will work in the case of our structure 5. We know that the interpretation of the S node is going to be a proposition, which will be true under some assignment function, e.g.

6. *Interpretation of S*
$[[S]]^a$ = 1 iff John offended $[[\mathbf{t_1}]]^a$.

(Remember, traces are just like pronouns!!)

We'll apply our P.A. rule to that denotation, and what we get is:

7. *Interpretation of S'*
a. $[\lambda x \quad D \text{ . } [[S]]^{a^{x/i}}]$.

and because we know that every element of S is assignment-independent except for the trace, we can rewrite 7a as 7b:

b. [λx D . John offended $[[\mathbf{t_1}]]^{a^{x/i}}$].

and by our traces and pronouns rule, that gives us:

c. [λx D . John offended x]

which is exactly a function of type <et>, which can happily combine with our DP **every linguist** (in its original, non-type-shifted meaning). When it does, what you get is:

d. The denotation of **every linguist** is:

$[\lambda f \quad D_{et} . [\lambda g \quad D_{et} .$ for every y s.t. f(y)=1, g(y) =1]]([λx D_e . x is a linguist])

= [λg D_{et} . for every y s.t. y is a linguist, g(y) =1]

e. So, combining the denotation of **every linguist** with the denotation of the S' node, given in c. above, we get:

$[\lambda f \quad D_{et}$. for every y s.t. y a is a linguist, f(y) =1]([λx D . John offended x])

=1 iff for every y s.t. y is a linguist, [λx D . John offended x](y) =1

= 1 iff for every y s.t. y is a linguist, John offended y.

which is the interpretation we want for the sentence, "John offended every linguist".

So, what happens when we do quantifier movement is that we create a structure that does PA over the constituent that the quantifier adjoins to. It's perhaps important to pay attention to what's doing the variable binding here: it's the index, which introduces the PA rule, which binds the variable and makes its interpretation assignment-independent, not the QP itself – although you'll often hear people talking about the QP 'binding' its trace. What they really mean is it's the *operator* (the index) which is introduced by QP movement which is actually doing the binding, and the QP is just then combining with an appropriate argument created by the binding in the usual way.

Who sees what's weird about this with respect to our analysis of relative clauses?
Then H&K go on to talk about what happens in successive-cyclic movement, whether intermediate traces creating predicate abstracts are created at every stop along the way. (Note that this type of theory about successive-cyclic movement has implications for the debate which some of you may remember from Howard Lasnik's talks about whether or not intermediate traces are deleted: if H&K are right, intermediate traces are crucial to interpretation, and may not be deleted.) Bob will tell us more about some potential uses for this intermediate-trace creation when he talks about Nissenbaum's paper on parasitic gaps.

5 Why Quantifier Movement?

5.1 Easy account of scope ambiguity:

8. The company sent one rep to every meeting

1= *For each meeting, there is a rep who went to it.*

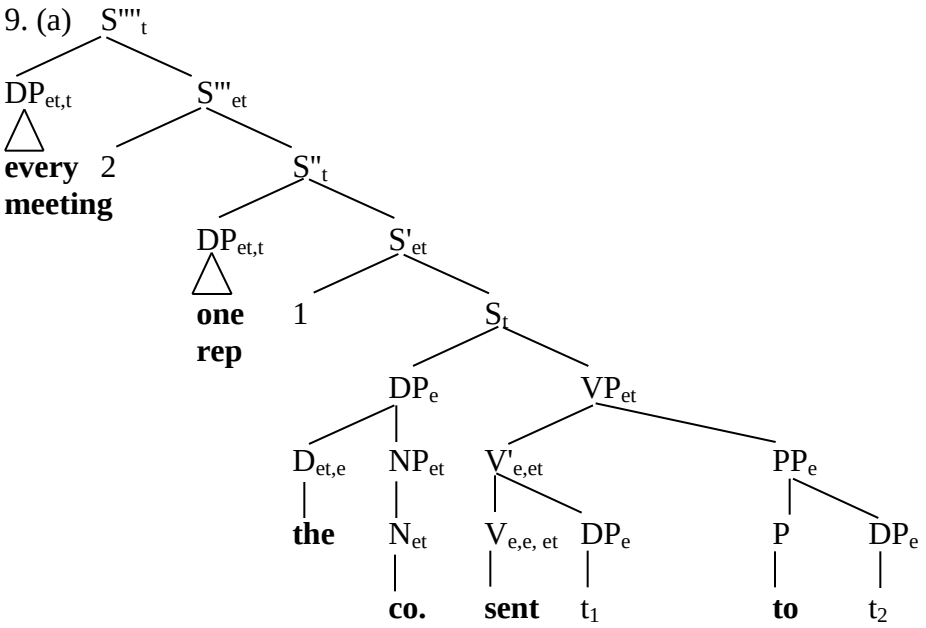
2= *There is one rep who went to all the meetings.*

(Recall that in predicate calculus, we could account for this type of ambiguity (at least in the **something** and **everything** cases) by simply reversing the order of the universal and existential

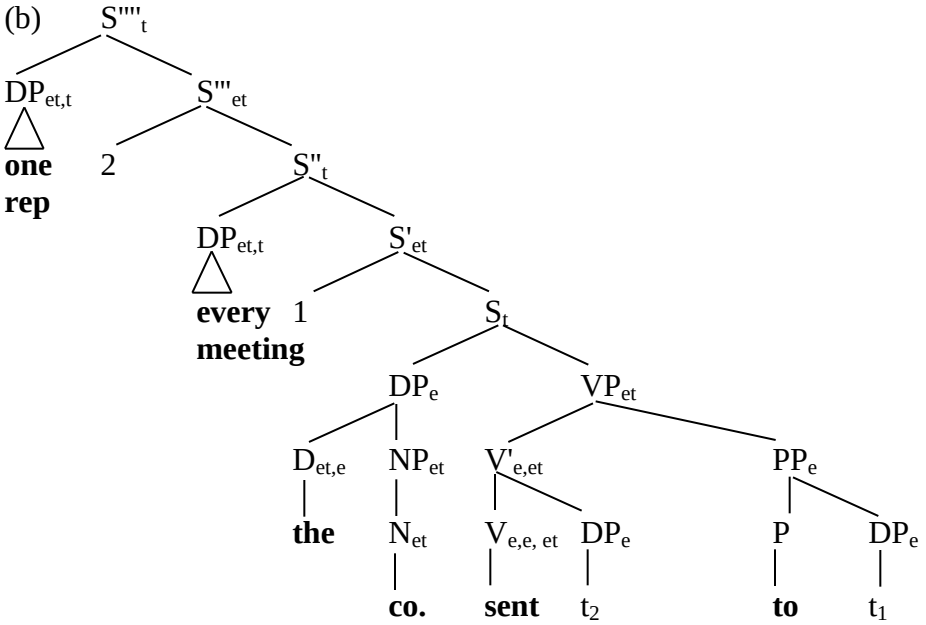
quantifiers. The trick was to write an interpretive rule that, given the single input (a), could produce both formulas).

Quantifier movement gives us an easy option: depending upon which quantifier moves first and which second, we'll get different structures. And for each structure, there'll be a unique interpretation with a particular scope. (Optionality of movement! rather counter to the Minimalist Program's goals, yes?)

9. (a)



(b)



H&K give the following scenario:
There's 1 company c

who has 2 representatives, r1 and r2
and there are three meetings: m1, m2 and m3.
c sent r1 to m1, r2 to m2 and m3, and nobody else to anything else.

In these circumstances ,we can prove that 9a is true and 9b is false with the definitions we've already got for **every** and **one** (= **some** with cardinality of the intersection = 1).

Let's just check that 9a is true in this scenario (in the text, H&K check that 9b is false).
We'll model this checking on the way H&K do their checking. First, we'll just look at the denotation of S''', sister to **every meeting**, which will simplify things; we'll just have to decide if all of m1, m2 and m3 satisfy the predicate that's denoted by S'''.

10.:
a. [[S''']] =

b. $[\lambda x \quad D_e$. [[S'']] $^{x/1}]$ =

c. $[\lambda x \quad D_e$. $[[S'']]^{x->1}]$=

d. $[\lambda x \quad D_e$. $[[\text{one rep}]]^{x->1}$ ($[[S']]^{x->1}$)] =

e. $[\lambda x \quad D_e$. for at least 1 y {r1, r2}, $[[S']]^{x->1}(y)=1]$ =

f. $[\lambda x \quad D_e$. for at least 1 y {r1, r2}, $[[S]]^{x->1^{y/2}}]$ =

g. $[\lambda x \quad D_e$. for at least 1 y {r1, r2}, $[[S]]^{x->1, y->2}$] =

h. $[\lambda x \quad D_e$. for at least 1y {r1, r2}, $[[VP]]^{x->1, y->2}$ (c)] =

i. $[\lambda x \quad D_e$. for at least 1y {r1, r2}, $[[\textbf{sent}]]^{x->1, y->2}$ $([[\textbf{t}_2]]^{x->1, y->2})([[\textbf{t}_1]]^{x->1, y->2})(c)]$ =

j. $[\lambda x \quad D_e$. for at least 1y {r1, r2}, $[[\textbf{sent}]]^{x->1, y->2}$ (y)(x)(c)] =

k. $[\lambda x \quad D_e$. for at least 1y {r1, r2}, c sent y to x]

Now, to find the truth value for S'''', we simply have to check whether for every possible value of x {m1, m2 and m3}, the truth condition k holds (because that's the definition of "every meeting" : $[\lambda f \quad D_{et}$ **.** for all x s.t. x {x: x a meeting}, f(x) =1].
For x = m1, k is true (c sent r1 to m1).
For x= m2, k is true (c sent r2 to m2)
For x = m3, k is true (c sent r2 to m3)
Therefore, [[S'''']] = 1.

So, QR provides a natural way to get the right reading for multiple-scope sentences like 8. The movement case looks even better when you consider examples (from May) like 11:

11. One apple in every basket is rotten.

If we force "every basket" to get narrow scope, which is what happens on a flexible types approach here, we get a meaning for this sentence that is very unnatural: there is one specific apple that is in every basket and it is rotten. (Don't ask how one apple can be in a bunch of different baskets! that's the reading the sentence must necessarily get).

If, on the other hand , we allow QR of "every basket", we get the right meaning: for every x, x a basket, at least one y, y an apple is in it and is rotten.

5.2 Antecedent-contained deletion

Without QR, sentences like 11 are very tricky to analyse.

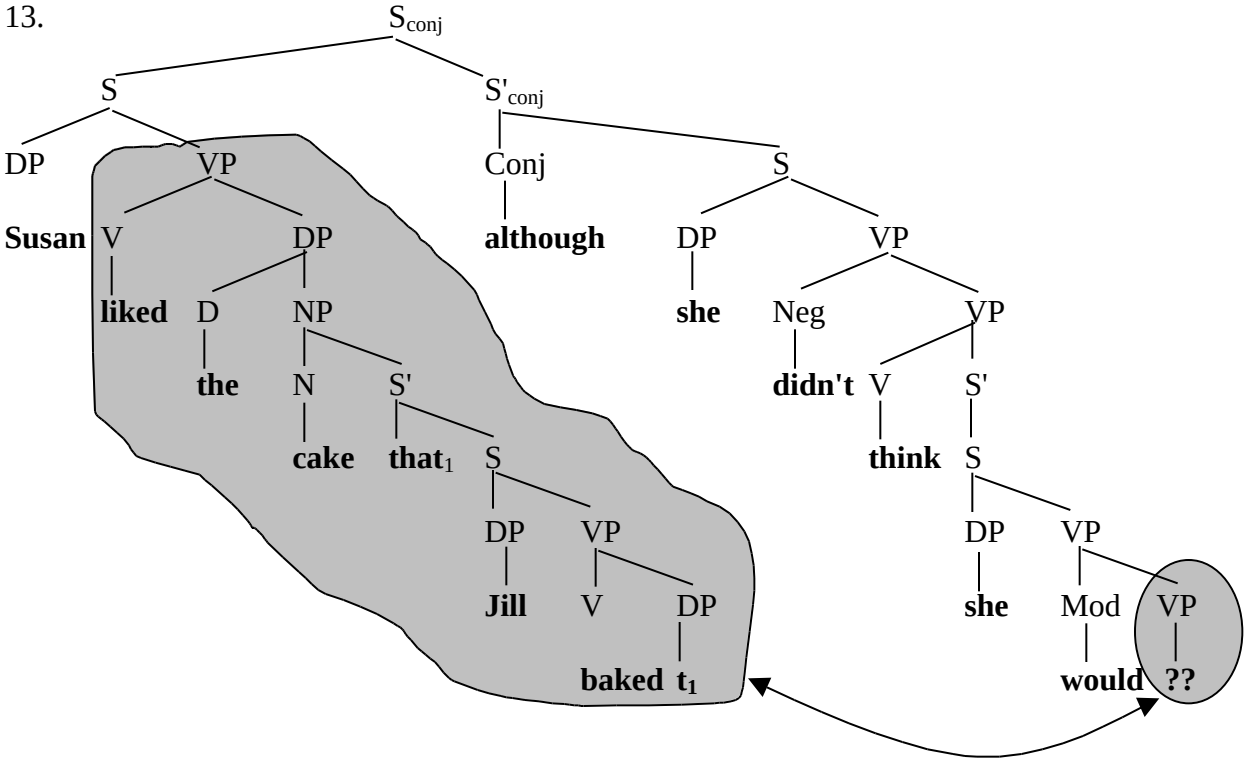
11. I asked Bill for every book that you didn't.

Here's the problem. VP-deletion is a very productive process in English. It seems that in general, we can omit VPs that are exact copies of VPs that have occured earlier in the sentence:

12. (a) I read *War and Peace* before you did.
(b) I went to Tanglewood although I wasn't supposed to.
(c) Susan liked the cake that Jill baked, although she didn't think she would.

Here's the structure for, e.g., (c):

13.



Now consider the necessary structure for 11:

14.