Introduction to Semantic Theory
Quantifying expressions II

Dr. Sarah Zobel

Class: June 29, 2016

Connecting back to the previous lecture

Central result: Sentences containing nominal quantifiers express statements about amounts/quantities of individuals.

- Nominal quantifiers express different kinds of relations that need to hold between three sets of individuals that are given either contextually, lexically, or explicitly in syntax.
- They are either atomic or composed (a quantificational determiner + an NP).
- Importantly: atomic and composed quantifiers behave the same way semantically!
- The type of nominal quantifiers is $\langle\langle e, t\rangle, t\rangle$.

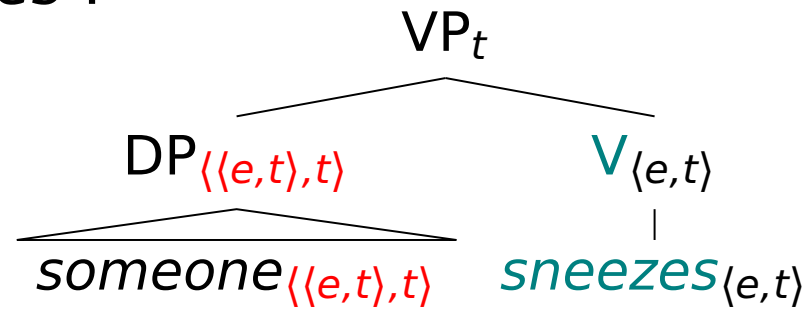
Aim for today

The aim for today: to propose an analysis for the treatment of nominal quantifiers in object position

- ⇒ Quantifiers in object position pose a new problem for the derivational system built up so far.
- ⇒ The types of solutions that were proposed in the case of attributive adjectives are both unattractive!
- ⇒ We will see: the same mechanism that is used to account for nominal quantifiers in object position can be used to account for the different readings in scope ambiguities and the interpretation of bound pronouns.

Recap: quantifiers in subject position

In the last lecture, we proposed the following structure for '*Someone sneezes*':

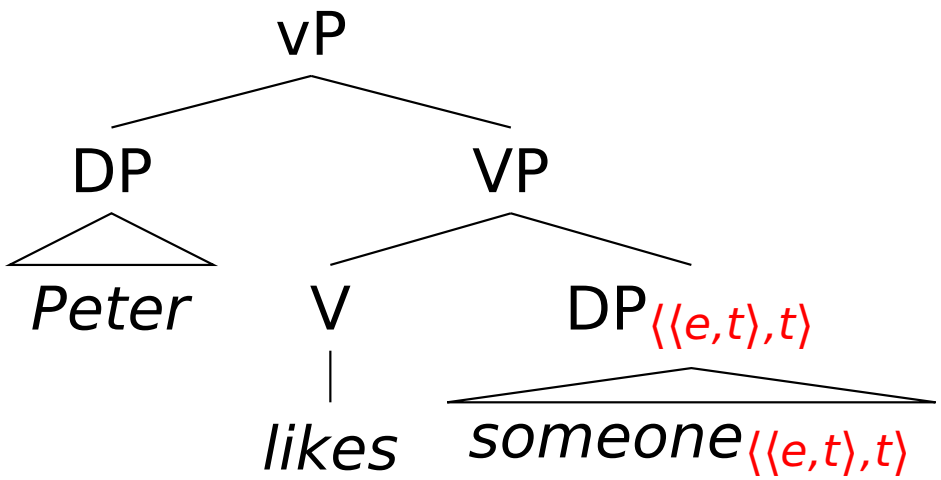


- The type of *someone*: $\langle\langle e, t\rangle, t\rangle$; it takes the denotation of its scope as its argument and outputs a truth-value.
- The final proposal for the extension of *someone*:

(1) $[\![someone]\!]^{w,g,c} =$
$\lambda P_{\langle e,t\rangle}.\exists x : x \in C$ & human'$(x)(w) = 1$ $[P(x) = 1]$

Object position?

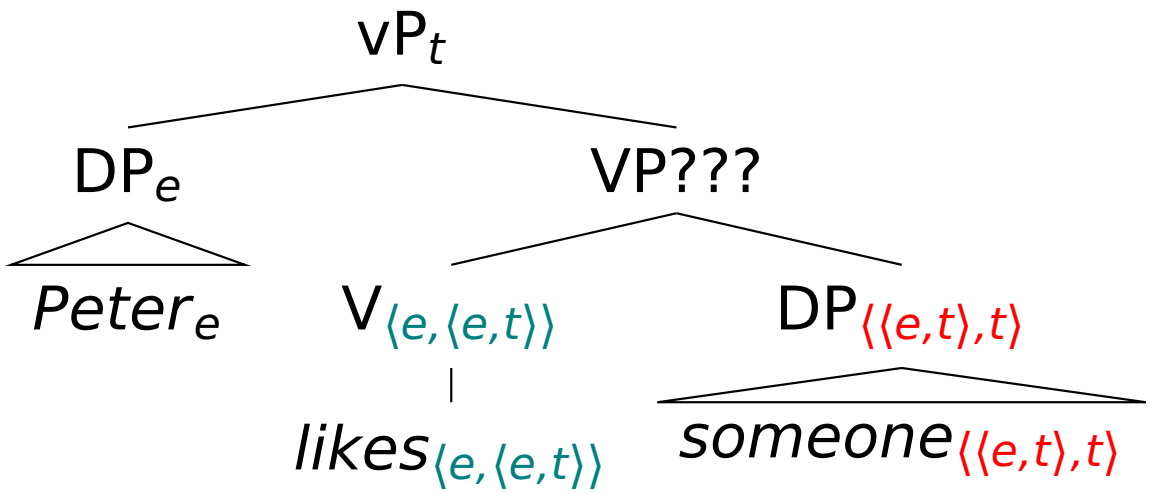
Quantifiers do not only occur in subject position; we need to check whether the analysis also fits quantifiers when they occur in object position:



Do the semantic types of the lexical elements in this tree fit?
Can the truth condition be computed?

Empirical problem: “object quantifiers”

The proposal for the type of quantifiers in subject position does not fit with what is needed compositionally for quantifiers in object position – neither (FA) nor (PM) can apply:



⇒ similar problem to the one with attributive adjectives

Possible solutions?

In which directions could possible solutions go?

Possible solutions?

In which directions could possible solutions go?

Taking the discussion on attributive adjectives as a model, we could either:

- change the types to make them fit.
- introduce another derivation rule for branching nodes with $\langle\langle e, t\rangle, t\rangle$ and $\langle e, \langle e, t\rangle\rangle$ as daughters.

Both are not attractive!

As we will see, these are not the only options that are available. Before we take a look at the third possibility: think about what we want to derive for the truth conditions.

Conceptual considerations – I

What does a sentence with *someone* or *everyone* in object position intuitively convey?

(2) a. *Peter likes someone.*
b. *Mary hates everyone.*

Try to describe in which cases these two sentences are true!

Conceptual considerations – II

Intuitively, sentences containing nominal quantifiers in object position also make statements about sets just like sentences containing quantifiers in subject position:

- For '*Peter likes someone*': the set of contextually given people contains at least one person who Peter likes.

  ⇒ For at least one person in the set of contextually given people it is the case that Peter likes him/her.

(3) $\exists x : x \in C \;\&\; \text{human'}(x)(w) = 1\; [\text{like'}(x)(\text{Peter})(w) = 1]$

Conceptual considerations – II

Intuitively, sentences containing nominal quantifiers in object position also make statements about sets just like sentences containing quantifiers in subject position:

- For '*Peter likes someone*': the set of contextually given people contains at least one person who Peter likes.

  ⇒ For at least one person in the set of contextually given people it is the case that Peter likes him/her.

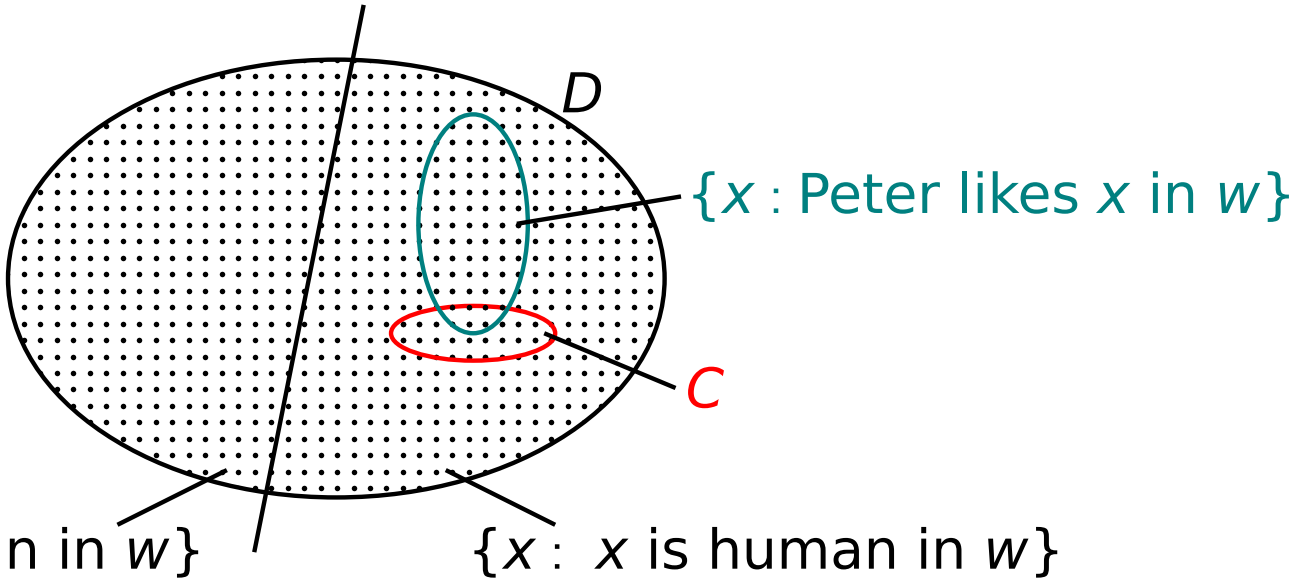
- For '*Mary hates everyone*': the set of contextually given people contains only people who Mary hates.

  ⇒ For every person in the set of contextually given people it is the case that Mary hates him/her.

(3) $\forall x : x \in C \;\&\; \text{human'}(x)(w) = 1$ [$\text{hate'}(x)(\text{Mary})(w) = 1$]

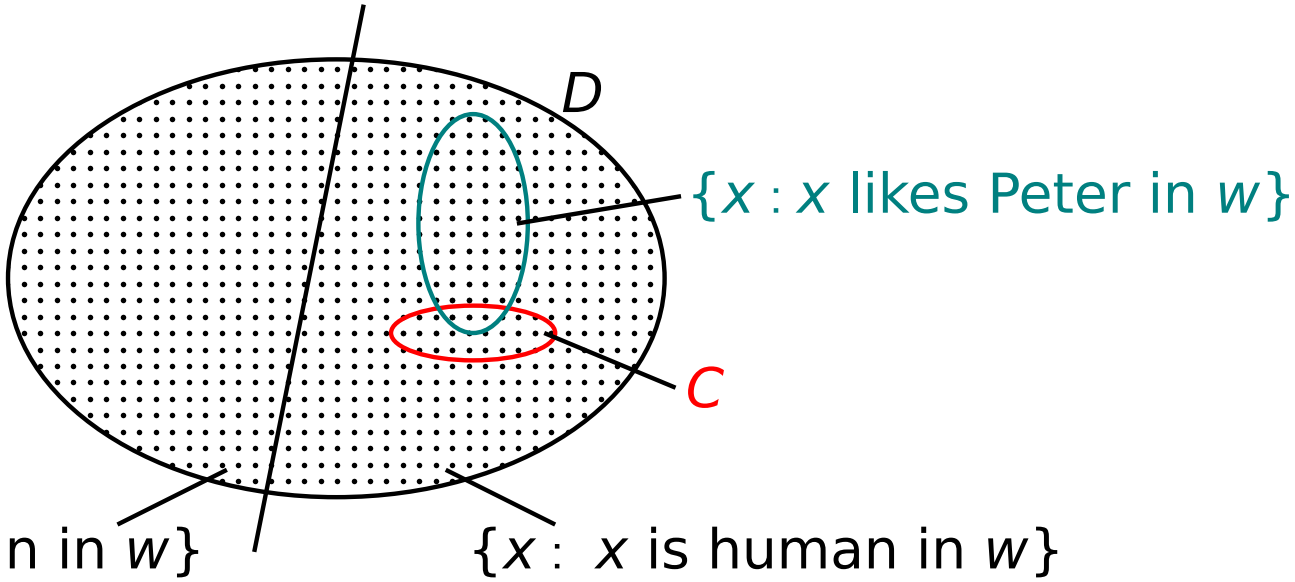
Conceptual considerations – III

The following illustration shows one set of circumstances in which '*Peter likes someone*' is true.



Conceptual considerations – IV

Compare the illustration for '*Peter likes someone*' to the following illustration for '*Someone likes Peter*'.



What we need:

Following the conceptual considerations, we take a closer look at '$\{x : \text{Peter likes } x \text{ in } w\}$'.

In function notation this is:

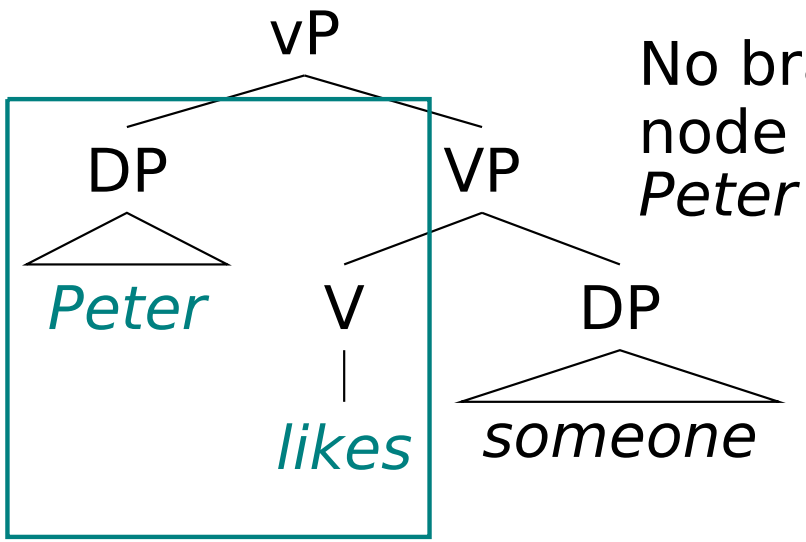
(4) $\lambda x_e.\ \text{Peter likes } x \text{ in } w = \lambda x_e.\ \text{like'}(x)(\text{Peter})(w)$

If we could derive this extension to give it as input to the regular extension of *someone* we would have a solution for the problem!

(5) $[\![\textit{someone}]\!]^{w,g,c} =$
$\lambda P_{\langle e,t\rangle}.\exists x : x \in C \ \&\ \text{human'}(x)(w) = 1\ [P(x) = 1]$

A new combinatoric problem – I

Since the lexical elements that feature in the extension (λx_e. Peter likes x in w) do not form a subtree in which *someone* is not included, it is not clear, how to derive this extension from what is given:



No branching node connects *Peter* and *likes*!

A new combinatoric problem – II

Also, even if we found a way to compose the extensions of *Peter* and *likes*, we would have a problem!

(6) a. $[\![\textit{Peter}]\!]^w = \text{Peter}$
b. $[\![\textit{likes}]\!]^w = \lambda y_e.\lambda x_e.\ \text{like'}(y)(x)(w)$

(7) $[\![\textit{likes}]\!]^w([\![\textit{Peter}]\!]^w)$
$= \lambda x_e.\ \text{like'}(\text{Peter})(x)(w)$ NOT WHAT WE WANT!

Peter should be the second argument of *likes* (= x); but we cannot fill the second argument unless the first argument has been filled!

Summary of the problem

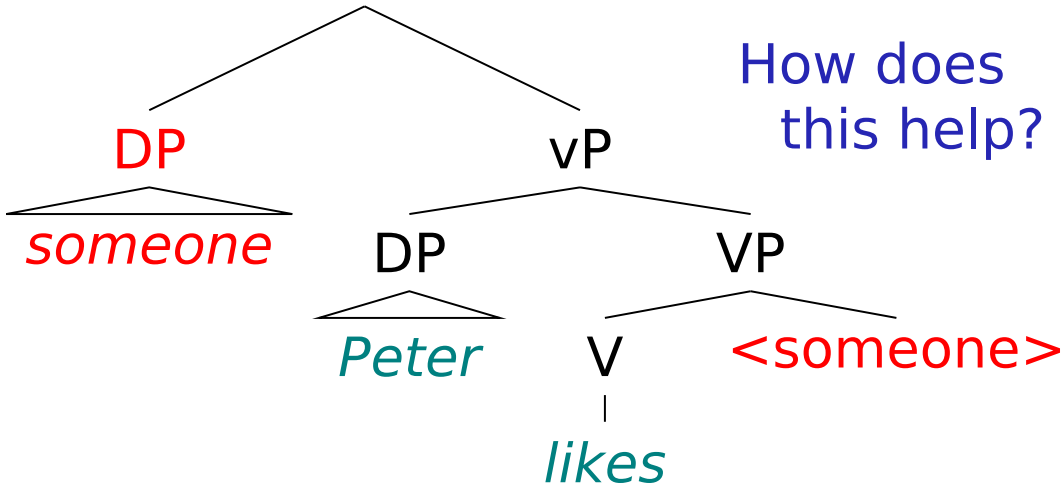
We run into two independent problems with quantifiers in object position:

- Considering the types, the type proposed for quantifiers in subject position does not work for quantifiers in object position. We get a type mismatch between the type of the verb and the type of the quantifier.
- Conceptual considerations tell us that the type and extension determined for the quantifiers in subject position can also be used for quantifiers in object position. However, it is unclear how to derive the desired input for the quantifier from the LF that we assumed and the given extensions.

Solution: raising the quantifier in object position

To derive the two readings of sentences containing two nominal quantifiers: assume quantifier raising (QR).

Nominal quantifiers in object position: use QR to avoid the type mismatch AND make the desired input for the quantifier derivable. The result of QR for '*Peter likes someone*':

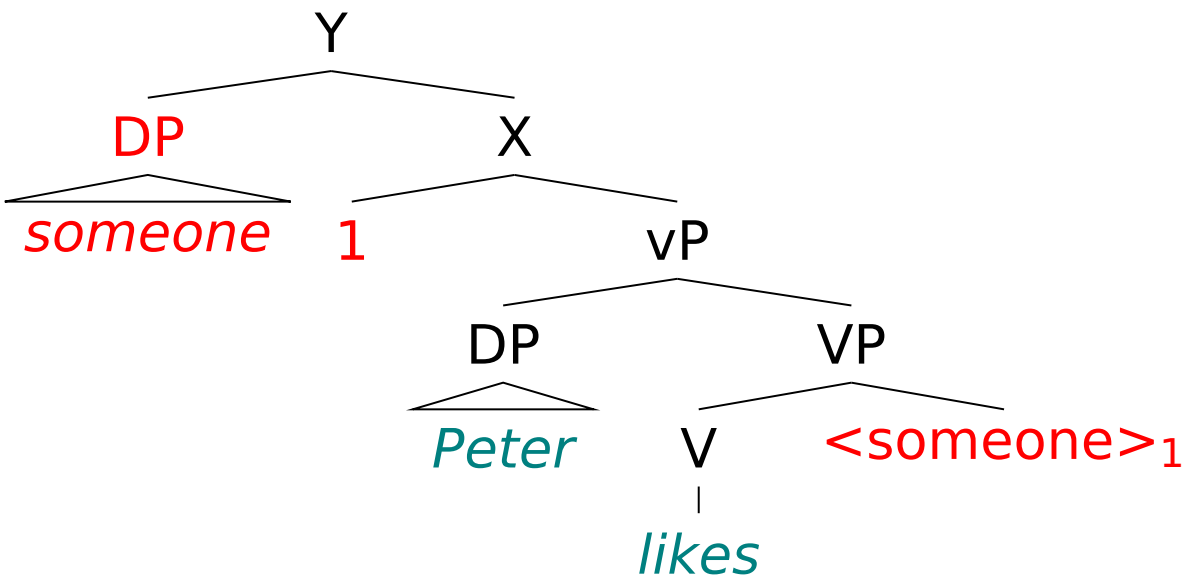


Interpreting LFs that result from QR

To get rid of the type mismatch and to make the desired input for *someone* derivable, we need two assumptions on how we should interpret the result of QR:

- In the position from which *someone* has moved, we find a trace/copy/ghost that needs to be interpreted. Semantic proposal: traces behave like pronouns.
- QR creates a kind of anaphora-like dependency between the moved quantifier and the trace. This dependency is expressed by introducing a binder-index into the LF.

The new LF for '*Peter likes someone*'



The labels *X* and *Y* are just dummy labels so that we can refer to the nodes when talking about the tree.

Interpreting the trace

To interpret the trace of the quantifier, we can extend the interpretational rule for pronouns into the pronouns and traces rule (PTR):

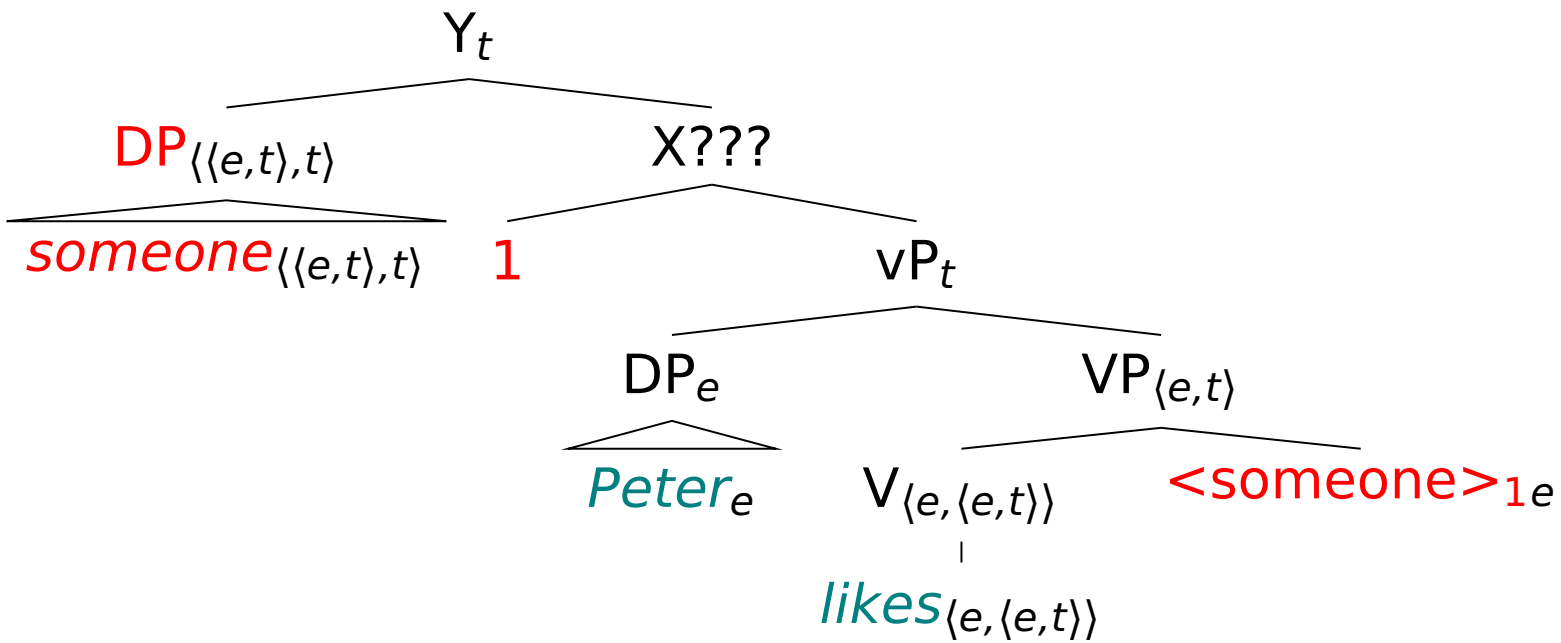
(8) Pronouns and Traces Rule (PTR):
If α is a pronoun or a trace, then for any assignment g and index i for which g assigns a value: $[\![\alpha_i]\!]^{w,g,c} = g(i)$

This means, of course, that traces are of type e.

This resolves the type mismatch inside vP!

The type-annotated new LF – I

The types inside the vP now fit:



What should be the type of the X-node?
How can we derive it?

The interpretation of binder-indices

The idea is that the binder-index triggers a new derivation rule: predicate abstraction (PA).

(9) Predicate Abstraction (PA):
For a branching node α with the set of daughters $\{\beta, \gamma\}$, where β is a binder-index i and γ is of type σ, then $[\![\alpha]\!]^{w,g,c} = \lambda x_e.[\![\gamma]\!]^{w,g[x/i],c}$ (a function of type $\langle e, \sigma \rangle$)

(10) $g[x/i]$ means:
the assignment function is changed so that the variable x is assigned to the index i.

Note: The binder-index has no type! It only triggers (PA).

Why a new rule?

We assume a new rule instead of assigning a certain type and extension to binder indices since the effect that triggering (PA) has cannot be assigned as an extension:

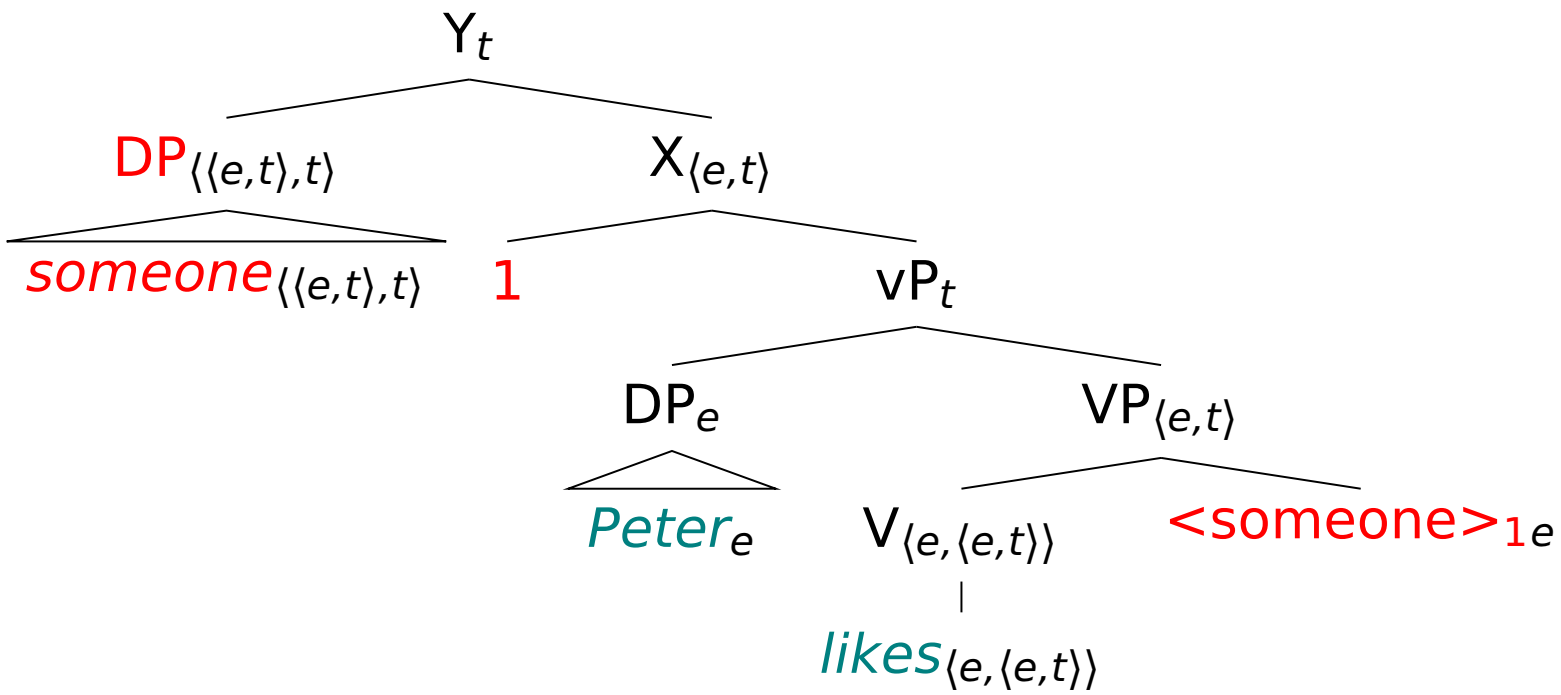
(11) $\lambda x_e.[\![\gamma]\!]^{w,g[x/i],c}$

The crucial point is that the assignment function g is modified. Since g is an interpretational parameter, simple extensions do not have access to it.

Note: we assume (PTR) for the same reason – instead of assigning an extension to pronouns and traces.

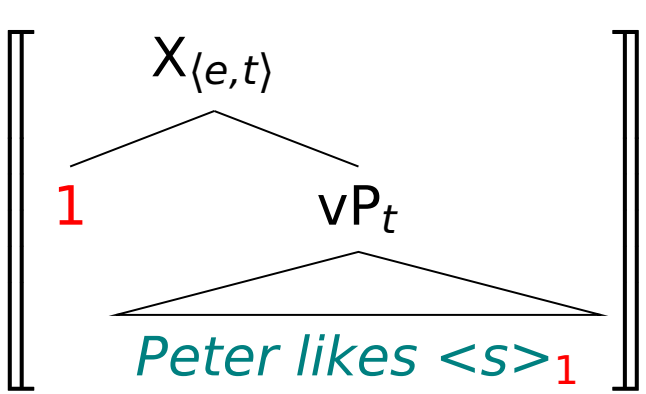
The type-annotated new LF – II

Application of (PA) at the branching node above the binder-index 1 results in:



Parts of a derivation: *Peter likes someone* – I

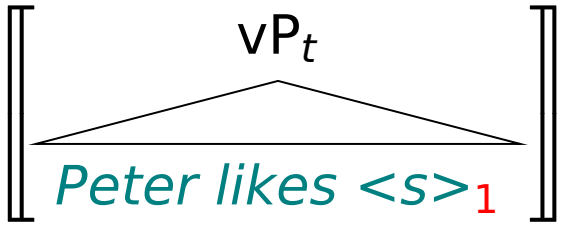
(12)



^{w,g,c}

$\overset{(PA)}{=} \lambda y_e.[\![\text{vP}]\!]^{w,g[y/1],c}$

(13)



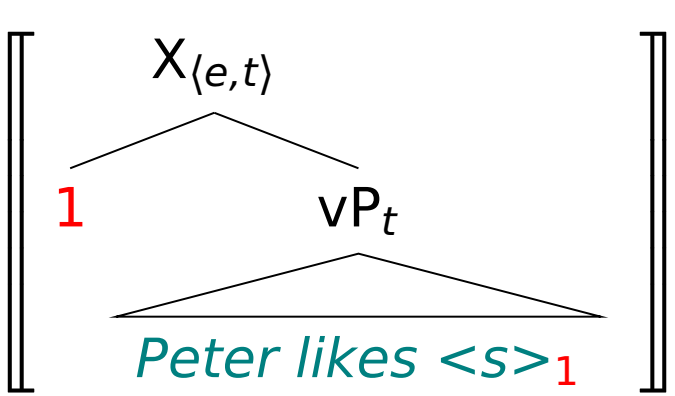
$^{w,g[y/1],c}$

$\overset{\text{a lot of steps}}{=}$ like'$(g[y/1](1))$(Peter)(w)

$=$ like'(y)(Peter)(w)

Parts of a derivation: *Peter likes someone* – II

Using the result of the "derivation" in (13):

(14) 

$$\overset{(PA)}{=} \lambda y_e.[\![\text{vP}]\!]^{w,g[y/1],c}$$

$$\overset{(11)}{=} \lambda y_e.\ \text{like'}(y)(\text{Peter})(w)$$

Using the new LF, (PA), and (PTR), we are able to derive that the extension of the X-node is the desired input for the quantifier *someone*! PROBLEM SOLVED!

Interim Summary

- The solution to the problem we encounter for quantifiers in object position is to assume QR for the quantifier – even if there is no ambiguity to resolve.
- To be able to interpret the resulting LF structure after QR, we need to extend the pronouns rule to (PTR) and add another derivation rule to deal with binder-indices:

(15) Predicate Abstraction (PA):
For a branching node α with the set of daughters $\{\beta, \gamma\}$, where β is a binder-index i and γ is of type σ, then $[\![\alpha]\!]^{w,g,c} = \lambda x_e . [\![\gamma]\!]^{w,g[x/i],c}$

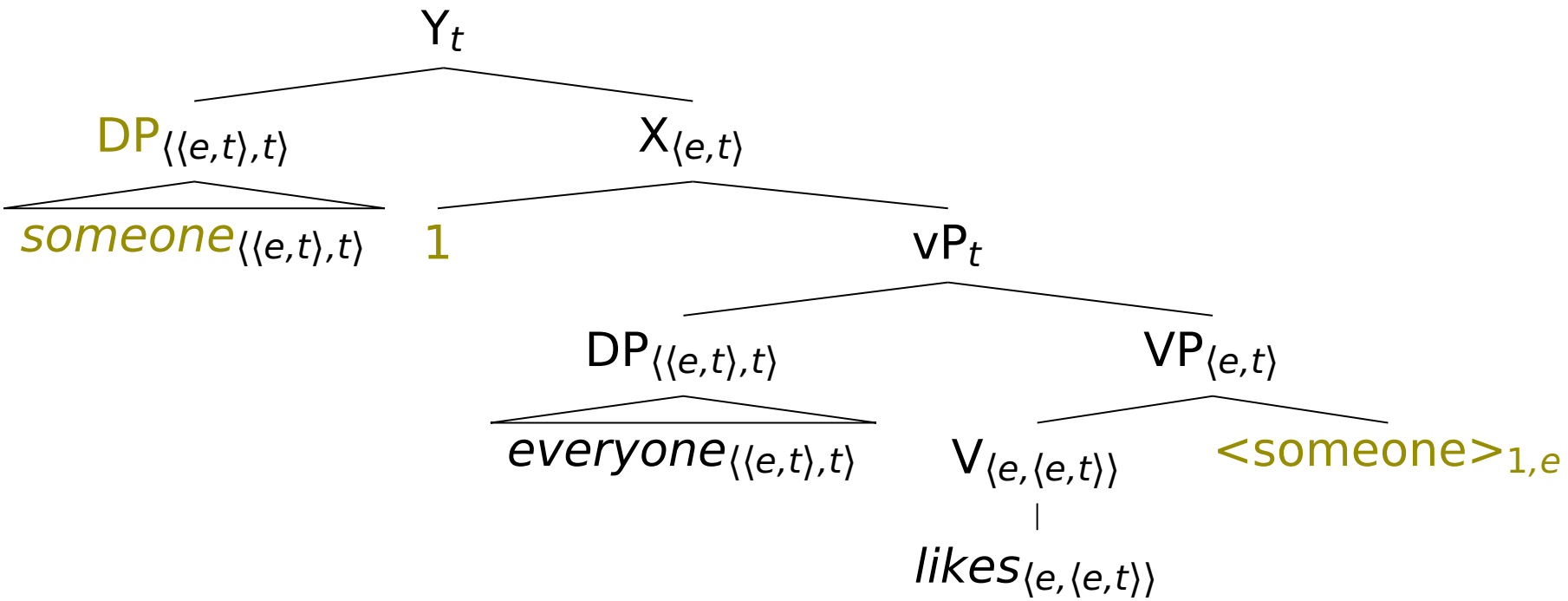
Combining subject and object quantifiers

We can now also derive the truth conditions of sentences that contain nominal quantifiers in subject and object position simultaneously:

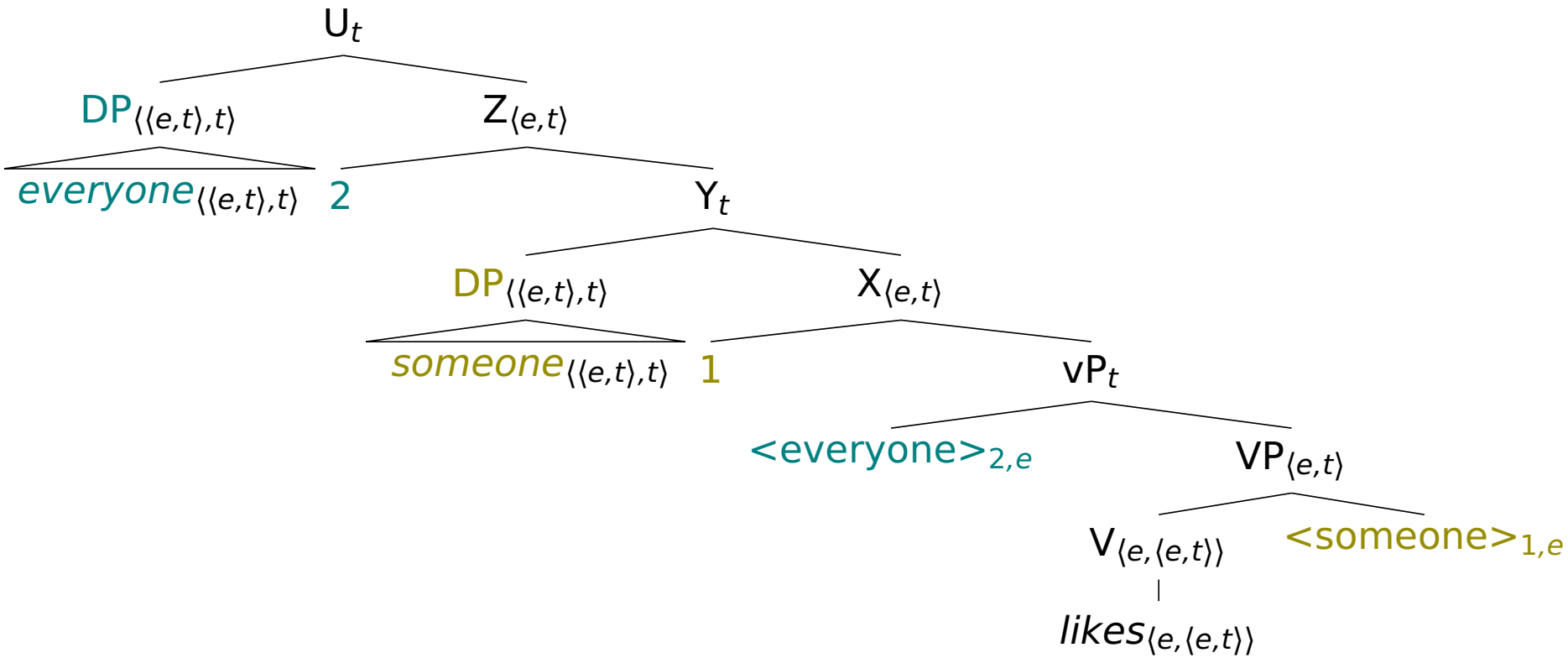
(16) *Everyone likes someone.*

Importantly: This sentence is ambiguous. This means – different LFs serve as input structure for the derivation of the truth conditions for each of the two readings.

Inverse scope reading: *Everyone likes someone*



Surface scope reading: *Everyone likes someone*



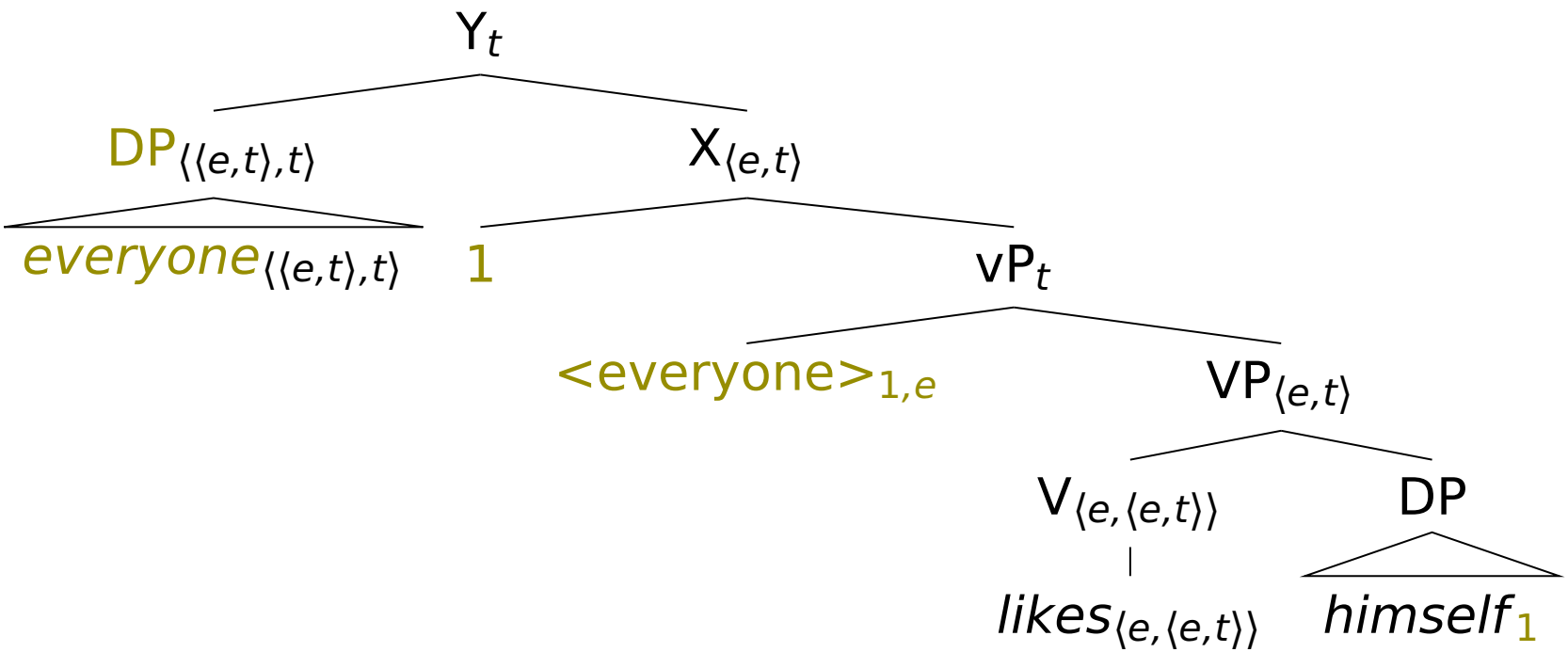
Accounting for bound pronouns

We have seen that the interpretation of third person singular pronouns can be dependent on a nominal quantifier in the same sentence:

(17) a. *No actress resents her cat.*
b. *Everyone likes himself.*

With the possibility of applying QR and the interpretation proposed for the resulting structure, it is easy to account for this covarying behavior by co-indexation of the trace and the pronoun.

LF: *Everyone likes himself*



Extension of the X-node: $\lambda z_e.\ \text{like'}(z)(z)(w)$

Summary

- Nominal quantifiers in object position pose a new problem for the derivational system built up so far.
- This problem can be solved by assuming and interpreting QR even in cases where no ambiguity needs to be resolved.
- Interpreting the result of QR requires the introduction of a new derivation rule (PA) and the amendment of the pronouns rule.
- Attractive consequences: to account for sentences with scope ambiguities and bound pronouns, we can use the same strategy and interpretation rules!