Background: the rules of interpretation, quantifiers, movement.

1. The most basic rules

In this class we will use the system of direct interpretation developed in (Heim & Kratzer 1998), where the interpretation is assigned to syntactic constituents via interpretation function designated by ⟦⟧.

(1) **Terminal nodes**
if α is a terminal node occupied by a lexical item, then ⟦α⟧ is specified in the lexicon

(2) **Non-branching nodes**
If α is a non-branching node and β is its daughter, ⟦α⟧=⟦β⟧

(3) **Functional application**
If α is a branching node and {β γ} is the set of its daughters then if ⟦⟦β⟧ is a function whose domain contains ⟦γ⟧, then ⟦α⟧ =⟦β⟧ (⟦γ⟧)

(4) **Predicate modification**
If α is a branching node and {β γ} is the set of its daughters and ⟦β⟧ and ⟦γ⟧ are both function of type <et> then ⟦α⟧ = λx_e. ⟦β⟧(x)=1 & ⟦γ⟧(x)=1.

A verb is interpreted as a function of type <et> shown in (6), for shortness I will write (7). The interpretation for a simple sentence 'Anna came' is derived as shown in (9).

(5)

(6) ⟦came⟧ = λx. 1 iff x came

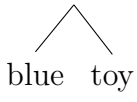
(7) ⟦came⟧ = λx. x came

(8) ⟦Anna⟧ = Anna

(9) ⟦Anna came⟧ = by FA
⟦came⟧ (⟦Anna⟧)= by TN
λx. 1 iff x came (Anna) = by lambda conversion
1 iff Anna came

The predicate modification rule is used to interpreted structures where multiple predicates are stacked on tope of each other, like shown in (10)

(10)



(11) [[blue]] = λz. z is blue

(12) [[toy]] = λy. y is a toy

(13) [[blue toy]] = by PM

λx. [[blue]](x) =1 & [[toy]](x)=1 = by TN

λx [λz. z is blue (x)] =1 & [λy. y is a toy (x)] =1 =2 applications of lambda conversion

λx. x is blue & x is a toy

2. Traces and pronouns, quantifier movement, predicate abstraction

2.1 Pronouns

(14) He came.

What is the meaning of 'he'? If I am pointing at John, he can refer to John. If I am pointing at Seth, 'he' can refer to Seth. It seems that the meaning of 'he' depends on the context and can change its meaning.

How can we implement this? So far in our system the interpretation is not sensitive to a context. We can change that. We will represent this by adding an index g on our interpretation function $[[]]^g$.

g will stand for a function that takes care of those expressions the meaning of which is not specified in the lexicon. We call this function the assignment function. This is a function that takes a pronoun and returns its reference.

Imagine that a conversation is a game, in which for each round we adopt special assumptions about the reference of pronouns.

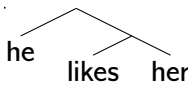
Let's assume that for our specific conversation, we adopted the function g shown in (15).

(15)

$$g := \begin{bmatrix} he \rightarrow John \\ she \rightarrow Mary \end{bmatrix}$$

(16) He likes her.

(17)



Now we need a special rule that tells us something like this: if you see a pronoun, check your g for its meaning

(18) **Pronouns Rule (the first attempt, to be revisited)**
If α is a pronoun, g is a variable assignment and $\alpha \in Dom(g)$, then $[\![\alpha]\!]^g = g(\alpha)$

Now, we have everything to interpret (16).

(19) $[\![\text{he likes her}]\!]^g$ = by FA
$[\![\text{likes her}]\!]^g ([\![\text{he}]\!]^g)$ = by FA
$[\![\text{likes}]\!]^g ([\![\text{her}]\!]^g) ([\![\text{he}]\!]^g)$ = by TN and PR
$\lambda x. \lambda y.$ y likes x (g(her)) (g(he)) =by lambda conversion and g
$\lambda y.$ y likes Mary (g(he)) = by lambda conversion and g
1 iff John likes Mary

Why did I say that the rule must be reconsidered? Imagine this sentence:

(20) He likes him.

Imagine, a conversation where the first 'he' refers to John, the second 'he' refers to Seth.

Now we are in troubles. g is a function. It cannot give a different output for one and the same input 'he'. How can we fix it? We need a way of distinguishing between the two occurrences of 'he'.

Here is what we will do. We will assume that pronouns come with invisible numerical indices.

(21) He_1 likes him_2.

We will change our g in such a way that it maps numerical indices into references.

(22)
$$g := \begin{bmatrix} 1 \rightarrow \textit{John} \\ 2 \rightarrow \textit{Seth} \end{bmatrix}$$

We will change our pronouns rule accordingly:

(23) **Pronouns Rule (the second attempt)**
If α_n is a pronoun, n is a numerical index, g is a variable assignment and $n \in Dom(g)$, then $[\![\alpha_n]\!]^g = g(n)$

(24) $[\![\text{he}_1 \text{ likes him}_2]\!]^g$ = by FA
$[\![\text{likes he}_1]\!]^g ([\![\text{him}_2]\!]^g)$ = by FA
$[\![\text{likes}]\!]^g ([\![\text{him}_2]\!]^g) ([\![\text{he}_1]\!]^g)$ = by TN and PR
$\lambda x. \lambda y.$ y likes x (g(2)) (g(1)) =by lambda conversion and g
$\lambda y.$ y likes Seth (g(1)) = by lambda conversion and g
1 iff John likes Seth

2.2 Quantifiers

2.2.1 The meaning of quantifiers

There are expressions that do not refer to any object or a group or a set of objects in the world. Let's consider the meaning of 'every student'. One initial hypothesis might be that 'every student' denotes a group consisting of all students. In this case, we predict that (25) should have a sensible meaning just like (26).

(25) #Every student outnumbers the professors.
(26) The class of students outnumbers the professors.

Moreover, we observe that (27) is ambiguous.

(27) Every Italian doesn't like Pavarotti.
- Meaning 1: Not every Italian likes Pavarotti.
- Meaning 2: Every Italian dislikes Pavarotti.

This is unexpected if 'every Italian' means 'the class of Italians', as shown by the unambiguous (28). (Chierchia & McConnell-Ginet 2000)

(28) The class of Italians does not like Pavarotti.

Let's look at another case:

(29) Some student admires every professor.

This sentence is ambiguous:

Meaning 1:
(30) One student is such that he admires every professor.
This is true in the following scenario: John is a student and he is very enthusiastic about this university. He admires every professor

Meaning 2:
(31) For every professor there is a student such that he admires him.
This is true in the following scenario: there are 5 students in a group overall and even though there is no one professor that everybody likes, each student admires at least one professor.

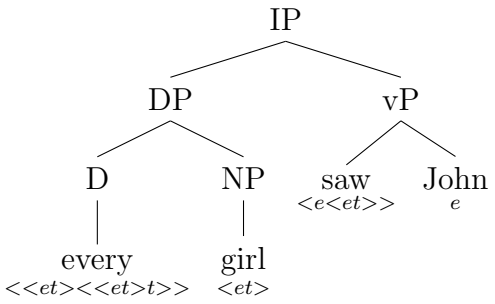
We call elements like 'every', 'some' quantifiers.

I will assume the following lexical entries for quantifiers:

(32) $[\![\text{some}]\!] = \lambda P_{<et>}.\lambda Q_{<et>}.\ \exists x[P(x)=1 \ \& \ Q(x)=1]$

(33) $[\![\text{every}]\!] = \lambda P_{<et>}.\lambda Q_{<et>}.\ \forall x[P(x)=1 \rightarrow Q(x)=1]$

(34)



(35) $[\![\text{every girl}]\!]^g$ = by FA

$[\![\text{every}]\!]^g$ ($[\![\text{girl}]\!]^g$) = by TN

$\lambda P_{<et>}.\lambda Q_{<et>}.\ \forall x[P(x)=1 \rightarrow Q(x)=1]$ ($\lambda z.$ z is a girl) = by lambda conversion

$\lambda P_{<et>}.\lambda Q_{<et>}.\ \forall x[[\lambda z.$ z is a girl (x)]$=1 \rightarrow Q(x)=1]$ = by lambda conversion

$\lambda Q_{<et>}.\ \forall x[x$ is a girl $\rightarrow Q(x)=1]$

(36) $[\![\text{saw John}]\!]^g$ = by FA

$[\![\text{saw}]\!]^g$ ($[\![\text{John}]\!]^g$) = by TN

$\lambda x.\ \lambda y.$ y saw x (John)= by lambda conversion

$\lambda y.$ y saw John

(37) $[\![\text{every girl saw John}]\!]^g$ = by FA

$[\![\text{every girl}]\!]^g$ ($[\![\text{saw John}]\!]^g$) = by (35) and (36)=

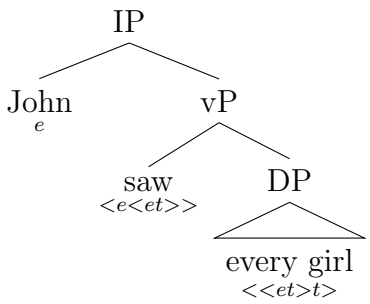
$\lambda Q_{<et>}.\ \forall x[x$ is a girl $\rightarrow Q(x)=1]$ ($\lambda y.$ y saw John) = by 2 lambda conversions

1 iff $\forall x[x$ is a girl $\rightarrow$ x saw John]

2.2.2 Quantifier raising

There is a problem with interpreting quantifiers in the object position.

(38)

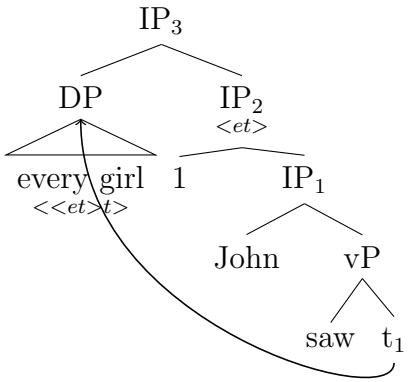


'Saw' and 'every girl' cannot compose by any of our rules. 'every girl' is looking for a predicate of type <et> . 'Saw' does not have the right type.

Here is what we will do. We will say that quantifier can undergo an invisible movement at the level of the logical form (LF). And we are going to design special rules to interpret this movement.

A quantificational DP like 'every girl' moves. It lives a trace of type e. A trace is like a pronoun. Think of this transformation as if we are turning this sentence into (40).

(39)



(40) Every girl is such that John saw her$_1$.

The rules we are going to develop will ensure that 'every girl' has a predicate of type <et> at its sister.

In order to interpret the resulting structure, we need to do two things: (1) modify out Pronouns rule in such a way that it can also apply to traces; (2) add a rule that allows us to interpret numerical indices like 1 in the structure. Those rules are given in (41) and (42).

(41) **Traces and Pronouns Rule**

If α_n is a trace or a pronoun, n is a numerical index, g is a variable assignment and n∈Dom(g), then $[\![\alpha_n]\!]^g$=g(n)

(42) **Predicate abstraction**

If α is a branching node and {β γ} is the set of its daughters, where β is a numerical index n, then for any variable assignment g, $[\![\alpha]\!]^g = \lambda x.\ [\![\gamma]\!]^{g(n->x)}$, where g(n->x) is a function that is just like g, but it assigns the value x to the numerical index n.

The idea is this: we fill the object argument of 'saw' with a variable and bind it creating a predicate.

(43) $[\![IP_2]\!]^g =$ by Predicate abstraction

$\lambda x.\ [\![IP_1]\!]^{g(1->x)}$ = by FA

$\lambda x.\ [\![vP]\!]^{g(1->x)}([\![John]\!]^{g(1->x)})$ = by FA

$\lambda x.\ [\![saw]\!]^{g(1->x)}([\![t_1]\!]^{g(1->x)})\ ([\![John]\!]^{g(1->x)})$ = by TN, T&P

$\lambda x.$ [$\lambda z.\ \lambda y.$ y saw z (g(1->x)(1)) (John)] = by 2 applications of lambda conversion and by g

$\lambda x.$ John saw x

(44) $[\![IP_3]\!]^g =$ by FA

$[\![$every girl$]\!]^g([\![IP_2]\!]^g)$ = by (35) and (43)

$\lambda Q_{<et>}.\ \forall x$[x is a girl → Q(x)=1] ($\lambda y.$ John saw y) = by 2 lambda conversions

1 iff $\forall x$[x is a girl → John saw x]

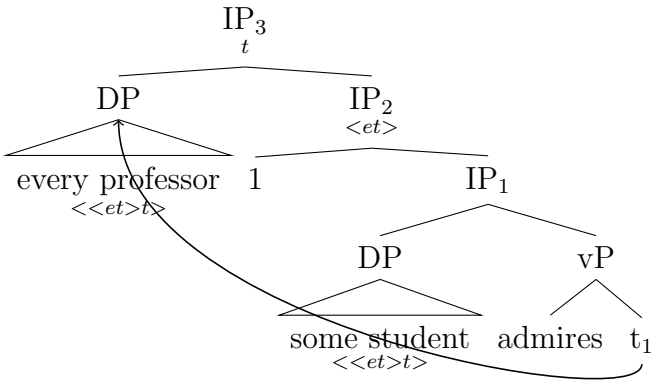
Now we have a handle on the ambiguity of (29) (repeated below as (45)).

(45) Some student admires every professor.

It can have two logical forms. Each meaning corresponds to one of the forms.

The inverse scope corresponds to the LF given in (46).

(46)



'Every professor' composes with the predicate created by abstraction. This predicate contains 'some student'.

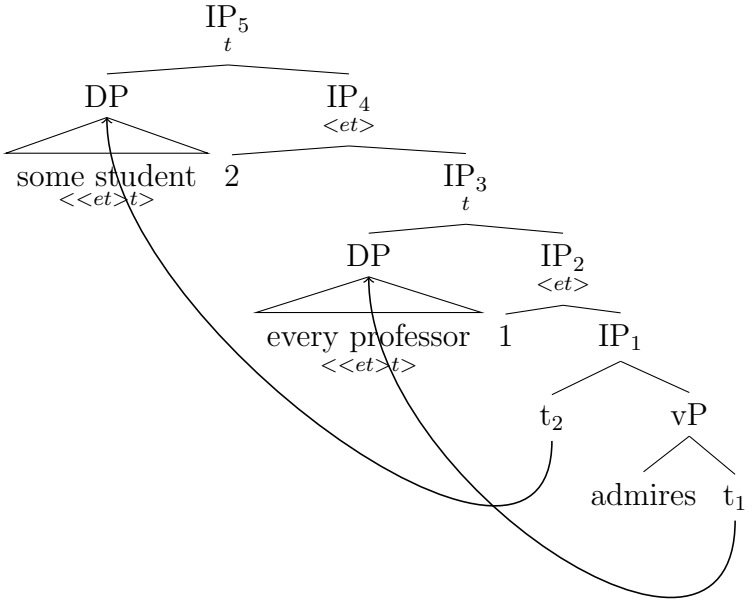
(47) $[\![IP_2]\!]^g = \lambda z_e.\ \exists x[x \text{ is a student } \& \text{ x admires z}]$

(48) $[\![(46)]\!]^g = 1$ iff $\forall y[y \text{ is a professor} \rightarrow \exists x[x \text{ is a student } \& \text{ x admires y}]]$

(49) Paraphrase: every professor is such that some student admires him

The direct scope is derived by two LF movements. The highest quantificational DP takes the widest scope in the sentence. The first LF movement is drive by the type-mismatch.

(50)



(51) $[\![IP_3]\!]^g = \lambda z_e.\ \forall y[y \text{ is a professor} \rightarrow z \text{ admires y}]$
(the set of things that admire every professor)

(52) $[\![(50)]\!]^g = 1$ iff $\exists x[x \text{ is a student } \& \forall y[y \text{ is a professor} \rightarrow x \text{ admires y}]]$

(53) Paraphrase: There is a student such that he admires every professor.

Literature:

Chierchia & McConnell-Ginet 2000, *Meaning and grammar*. Cambridge: MIT Press
Heim & Kratzer 1998, *Semantics in generative grammar*. Oxford: Blackwell