10 Syntactic Structure and Semantic Scope

10.1 Object Quantifiers and Quantifier Raising

10.1.1 Object Quantifiers and Scope Ambiguity

So far we were concerned with sentences that contain just one quantifier, a quantifier that also occurs in just one position, the subject position. But quantified NPs may occur in a variety of other positions, e.g. the object position, and we may have more than one quantified NP in one clause:

(1) a. Some girl liked every book.
b. No girl liked every book.

A problem with the derivation of (1) is that we have analyzed quantifiers as being of type $D_{(et)t}$, and transitive verbs as being of type D_{eet}. Obviously, these two meanings don't go together — we cannot use functional application here.

Another issue that appears when we consider sentences with more than one quantifiers is that such sentences often are ambiguous. For example, (1.a) has one reading in which it is stated that there is some girl x such that x read every book, and another reading that says that for every book y it holds that some girl read y. The latter reading is perhaps not very prominent, but it clearly exists for examples like the following:

(2) Some girl or other liked every book.

When we translate (1.a) into predicate logic, we can express the two readings as follows:

(3) a. $x[G(x) \quad y[B(y) \rightarrow L(x, y)]]$

b. $y[B(y) \rightarrow \quad x[G(x) \quad L(x, y)]]$

That is, the subject quantifier can have scope over the object quantifier, or the object quantifier can have scope over the subject quantifier. Often the reading with the subject quantifier having wide scope (over the object quantifier) is more natural, but there are many cases in which only the other interpretation makes sense:

(4) a. A nanny brought most kids to the kindergarten.
b. An American flag stood in front of every building.
c. A member of the Expose the Right group attended every public event held in Iowa or New Jersey. (The New Republic, Feb. 19, 1996).

We find such **scope ambiguities** with other operators as well. The following sentence can either mean that nothing that glitters is gold (wide-scope interpretation of *all*), or that it is not the case that everything is gold (wide-scope interpretation of negation). Of course, the idiom wants to express the second reading.

(5) All that glitters isn't gold.
a. $x[Gl(x) \rightarrow \neg Go(x)]$
b. $\neg \quad x[Gl(x) \rightarrow Go(x)]$

Let us see how we can deal with object quantifiers and how we can express such scope ambiguities.

10.1.2 Object Quantifiers by Quantifier Raising

In section (8) we were concerned with the phenomenon that a quantifier can bind a pronoun, and we have proposed to deal with this phenomenon by a movement operation for quantifiers. To be specific, we have assumed the following rule:

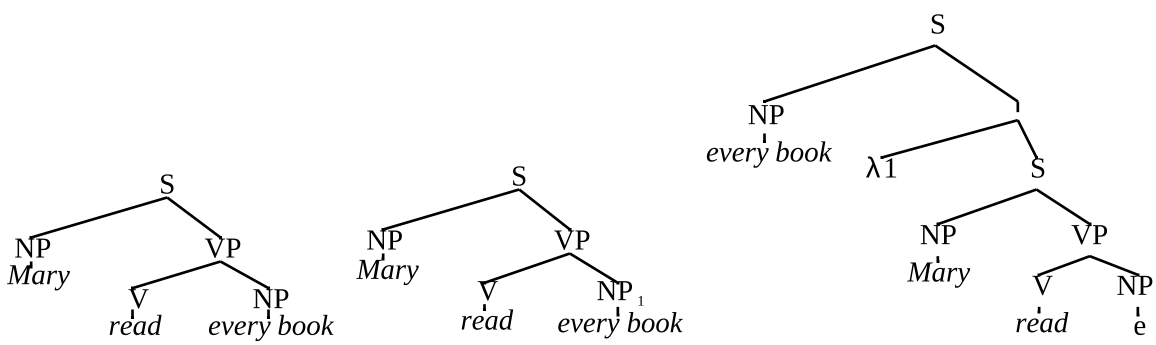
(6) If $[_S ... \alpha_{NP,i} ...]$ is a sentence that contains an NP α with index i,
then we can form the structure $[_S \alpha_{NP}\ \lambda i[_S ... e_i ...]]$ as input for semantic interpretation.

And we called the resulting structure **Logical Form** (LF) and distinguished it from Surface Structure (S-Structure). Recall the interpretation rule for structures like $\lambda i[_S ... e_i ...]$:

(7) $[\![\lambda i[_S \alpha]]\!] = \lambda g\ i\ \ \mathrm{DOM}(g), g\ \ \mathrm{DOM}([\![\alpha]\!])\ \ \lambda x\ \ D_e[[\![\alpha]\!](g[i/x])]$

This means that any trace or pronoun with the index i in α will be interpreted as x, which in turn creates a lambda term.

Now, the rule for quantifier raising works for subject quantifiers as well as for object quantifiers. This allows us to treat cases with object quantifiers, as follows:



S-Structure **Indexation** **Quantifier Raising (LF)**

The Logical Form then can be interpreted in the following way:

(8) a. LF: $[_S [_{NP} [_{Det}\ \textit{every}] [_N\ \textit{book}]]\ \lambda 1[_S [_{NP}\ \textit{Molly}] [_{VP}\ \textit{read}\ [_{NP}\ e]_1]]]$
b. $[\![[_{NP} [_{Det}\ \textit{every}] [_N\ \textit{book}]]]\!] = \lambda g \lambda P\ \ D_{et}[\lambda x[x \text{ is a book}]\ \ \ P], = ①$
c. $[\![\lambda 1[_S [_{NP}\ \textit{Molly}] [_{VP}\ \textit{read}\ [_{NP}\ e]_1]]]\!]$
$= \lambda g\ 1\ \ \mathrm{DOM}(g)\ \ \lambda x[\text{Molly read } g[1/x](1)],$
$= \lambda g\ 1\ \ \mathrm{DOM}(g)\ \ \lambda x[\text{Molly read } x], = ②$
d. $[\![[_S [_{NP} [_{Det}\ \textit{every}] [_N\ \textit{book}]]\ \lambda 1[_S [_{NP}\ \textit{Molly}] [_{VP}\ \textit{read}\ [_{NP}\ e]_1]]]]\!]$
e. $= \lambda g\ g\ \ \mathrm{DOM}(①), g\ \ \mathrm{DOM}(②)\ [①(g)(②(g))]$
f. $= \lambda g\ 1\ \ \mathrm{DOM}(g)\ [[\lambda P\ \ D_{et}[\lambda x[x \text{ is a book}]\ \ \ P](\lambda x[\text{Molly read } x])]$
g. $= \lambda g\ 1\ \ \mathrm{DOM}(g)\ [\lambda x[x \text{ is a book}]\ \ \ \lambda x[\text{Molly read } x]]$

This is the right result; it maps every assignment g (with 1 in its domain) to 1 if the set of books is a subset of what Molly read, and else to 0.

We see that quantifier raising leads to a structure in which the object argument is a trace, of type e. If transitive verbs are of type eet, then expressions like $[_{VP}$ *read* $[_{NP}e]_1]$ can be interpreted without any problem, by functional application:

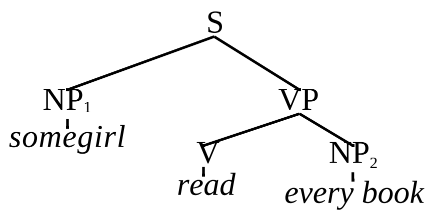
(9) $[\![[_{VP}\ \textit{read}\ [_{NP}e]_1]]\!] = \lambda g[[\![\textit{read}]\!](g)([\![e_1]\!](g))]$

10.1.3 Quantifier Raising and Scope Ambiguity

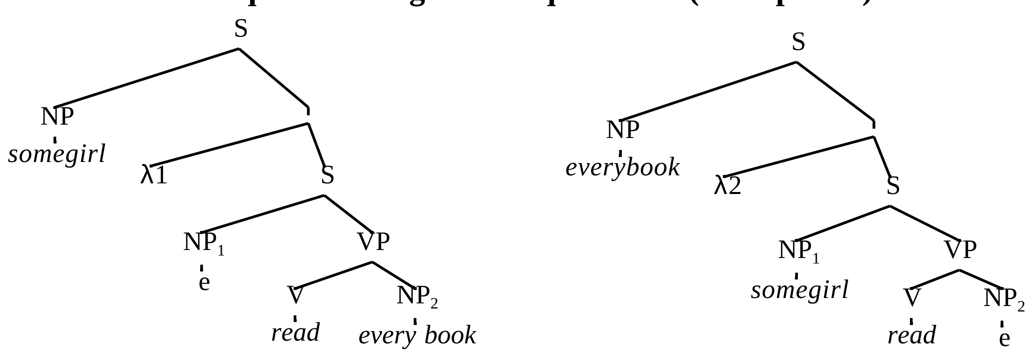
Quantifier raising also allows us to treat scope ambiguities in case of sentences with two quantifiers. Notice that the raising rule (6) can be applied more than once if necessary. It does not say anything about the order in which different applications of the rule should be executed. This allows for the expression of scope ambiguities:

(10)

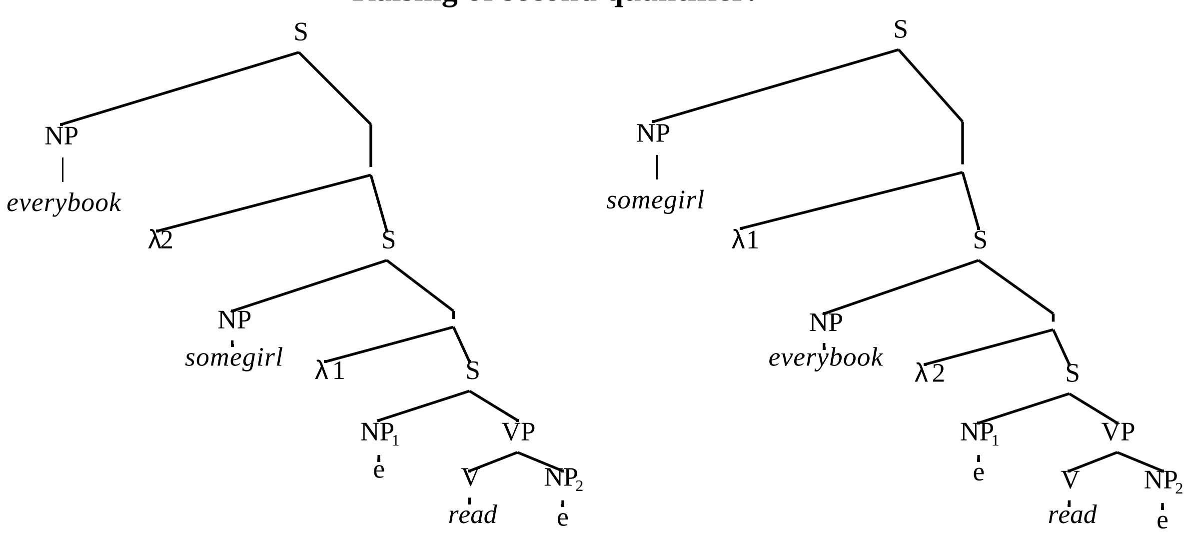
Indexed S-Structure:



Subsequent raising of first quantifier (two options):



Raising of second quantifier:



Consider the way how the left-hand structure is interpreted:

(11)a. $[\![[_S [_{NP} e]_1 [_{VP} [_V \textit{read}] [_{NP} e]_2]]]]\!]$
$= \lambda g\ 1,2\ \ \mathrm{DOM}(g)\ [g(1) \text{ read } g(2)]$, = ①

b. $[\![\lambda 1 [_S [_{NP} e]_1 [_{VP} [_V \textit{read}] [_{NP} e]_2]]]]\!]$
$= \lambda g\ \ 1\ \mathrm{DOM}$(①), $g\ \ \mathrm{DOM}$(①) λx[①$(g[1/x])$],
$= \lambda g\ 1,2\ \ \mathrm{DOM}(g)\ \ \lambda x[x \text{ read } g(2)]$, = ②

c. $[\![[_S [_{NP} \textit{some girl}]\ \lambda 1 [_S [_{NP} e]_1 [_{VP} [_V \textit{read}] [_{NP} e]_2]]]]]\!]$
$= \lambda g\ g\ \ \mathrm{DOM}$(①), $g\ \ \Delta\mathrm{OM}$(②) ($[\![\textit{some girl}]\!](g)$(①$(g)$)],
$= \lambda g\ 1,2\ \ \mathrm{DOM}(g)\ [\lambda x[x \text{ is a girl}] \cap \lambda x[x \text{ read } g(2)] \neq\ \ ]$, = ③

d. $[\![\lambda 2 [_S [_{NP} \textit{some girl}]\ \lambda 1 [_S [_{NP} e]_1 [_{VP} [_V \textit{read}] [_{NP} e]_2]]]]]\!]$
$= \lambda g\ \ 2\ \mathrm{DOM}(g),\ g\ \ \mathrm{DOM}$(③) λy[③$(g[1/y])$],
$= \lambda g\ 1,2\ \ \mathrm{DOM}(g)\ \ \lambda y[\lambda x[x \text{ is a girl}] \cap \lambda x[x \text{ read } y] \neq\ \ ]$, = ④

e. $[\![[_S [_{NP} \textit{every book}]\ \lambda 2 [_S [_{NP} \textit{some girl}]\ \lambda 1 [_S [_{NP} e]_1 [_{VP} [_V \textit{read}] [_{NP} e]_2]]]]]]\!]$
$= \lambda g\ \ \mathrm{DOM}$(④)$[[\![\textit{every book}]\!](g)$(④$(g)$)]
$= \lambda g\ 1,2\ \ \mathrm{DOM}(g)\ \ [\lambda y[y \text{ is a book}]\ \ \ \lambda y[\lambda x[x \text{ is a girl}] \cap \lambda x[x \text{ read } y] \neq\ \ ]]$

We get the right result, a function from assignments g that maps g to 1 if the books are a subset of the things that were read by a girl, and false otherwise.

10.2 Evidence for Quantifier Raising

In this section we will discuss some additional evidence for the idea that quantifiers undergo raising on a syntactic level of logical form.

10.2.1 VP-Ellipsis and Antecedent-Contained Deletion

In section (8) we have discussed the phenomenon of VP-ellipsis, illustrated by the following examples:

(12)a. Leopold read the letter, and Molly did, too.
b. Leopold read the letter before Molly did.

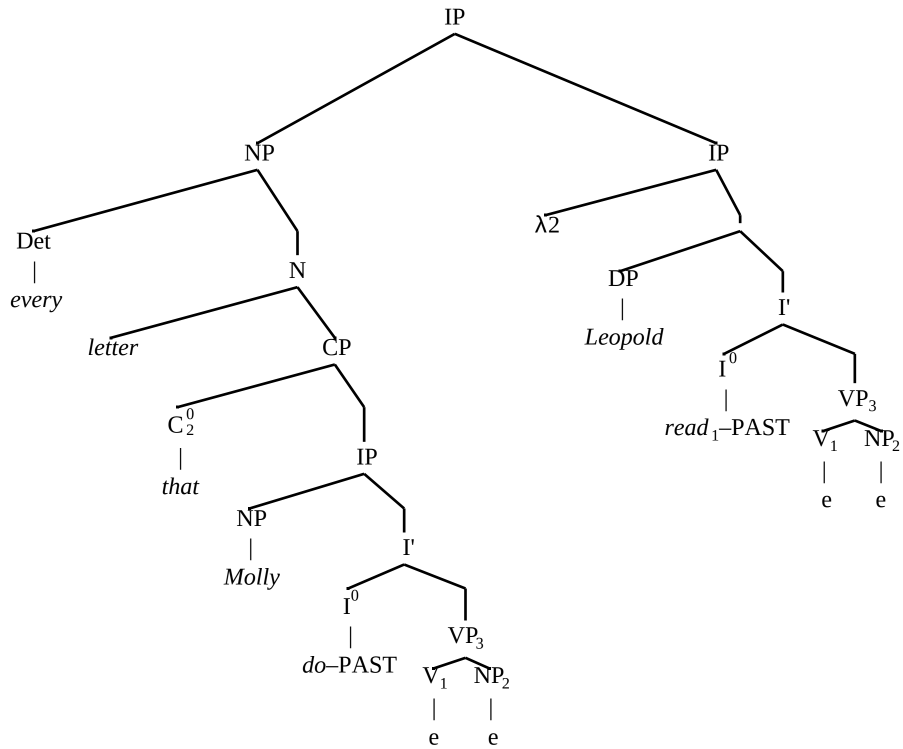
Now, consider the following case, which exhibits what is known as **antecedent-contained deletion**:

(13)Leopold read [$_{NP}$ every letter [$_{CP}$ that Molly did]].

What is remarkable here is that VP ellipsis occurs inside the object NP, which is *every letter that Molly did*. It has to be spelled out, of course, as ‘every letter that Molly read’. But notice that there is no antecedent VP that just consists of the verb *read* here; what we have is an antecedent *read every letter that Molly did*. But this certainly is not the proper antecedent; if we spell out *did* in (13) by *read every letter that Molly did* we get *before Molly read every letter that Molly did*. This cannot be the right result, of course, not the least because it again contains a VP ellipsis.

But see what happens is we allow the object NP to undergo Quantifier Raising:

(14)



Here, the NP *every letter that Molly did* has undergone Quantifier Raising. Its VP has to be interpreted just like its antecedent, which is the VP in the main clause that dominates two empty elements, and which have to be spelled out as [*read* e_2]. The trace e_2 is the variable that the quantifier *every letter that Molly did* quantifies over. This quantifier also quantifies over the trace e_2 in the antecedent, which we derived by relative clause formation. As a result we get the following interpretation:

(15)For every letter x such that Molly read x, Leopold read x.

This is the correct interpretation. Notice that it is crucial for this solution that the object quantifier, *every letter that Molly did*, undergoes Quantifier Raising. Hence the operation of Quantifier Raising, which was originally motivated to deal with the scoping of quantifiers, solves the problem of antecedent-contained deletion.

10.2.2 Syntactic Restrictions and Semantic Scope

The hypothesis of a syntactic representation level called Logical Form has been particularly attractive because it seems that the laws that govern extraction and raising of quantifiers are the same laws that govern the "visible" extraction and raising of certain expressions, such as wh-words, in Surface Structure. Only the existence of common restrictions of this type allow us to call LF a **syntactic** representation level.

This can be illustrated with the following examples. First, let us look at overt syntactic movement. There is a clear constrast of acceptability between examples (16.a-c) and (d):

(16)a. What$_1$ did John read e$_1$?
b. What$_1$ does Mary think [that John read e$_1$]?
c. What$_1$ did Mary read a book [about e$_1$]?
d. *What$_1$ did Mary read a book [that is about e$_1$]?

Extraction out of an object position (16.a), extraction out of a *that*-clause embedded by certain verbs (b), and extraction out of certain PPs (c) are possible; extraction out of a relative clause (d) is not. Hence relative clauses are **syntactic islands**. Compare these syntactic facts with the following semantic data:

(17)a. John read every article on Quantifier Raising.
b. Mary thinks that John read every article on Quantifier Raising.
c. Mary read a book about every famous French painter.
d. Mary read a book that is about every famous French painter.

Note that the sentences (17.a)-(c) have readings in which the quantified NP has wide scope. This is obvious for (a), as there is no other scope-bearing element around (names are, in a sense, scope-less). In predicate logic we can describe the interpretation of (a) as follows:

(17′) a. For every article x, Mary read x.
x[article(x) → read(m, x)]

Sentences (b) and (c) are actually scopally ambiguous in ways we haven't discussed so far but which are quite evident. As for (b), *think* acts as a scope-inducing element. We can state the readings as follows:

(17′) b.i. For every article x, Mary thinks that John read x.
x[article(x) → thinks(m, read(j, x))]

b.ii. Mary thinks that the following holds: For every article x, John read x.
thinks(m, x[article(x) → read(j, x)])

For reading (i) Mary might not be aware that the articles she thinks John has read are actually all the articles. In (ii), it is part of the content of Mary's belief that John read EVERY article. We now turn to (c).

(17′) c.i. For every painter x there is a book y about x and Mary read y.
x[painter(x) → y[book(y) about(y, x) read(m, y)]]

c.ii. There is a book y such that for every painter x, y is about x, and Mary read y.
x[book(x) y[painter(y) → y is about x] read(m,x)]

The interesting thing is that sentence (d) is **not** ambiguous in the same way as (b) or (c) are. That is, we only have a reading like (c.ii); the reading corresponding to (c.i) is unavailable:

(17′) d.i. (= c.i): absent.
d.ii. (= c.ii): available.

It is tempting to explain the absence of reading (d.i) using the same principle that we employed to explain why (1.d) is bad. The surface structure of (16.d) and the logical form of (17′.d.i) share the property that an expression (a wh word, or a quantified NP) is extracted out of a relative clause:

(18)a. *What$_1$ did Mary read a book [that is about e$_1$]?
b. *[$_S$[$_{NP}$every writer]$_1$ [$_S$Mary read a book [that is about e$_1$]]]

Note, in particular, that (17′.d.i) is not semantically odd at all -- we can express it easily, see (c.i). The assumption is that (17′.d.i) is ruled out syntactically. Hence LF is seen as a syntactic level of representation, following the same rules as overt syntactic constructions that we can observe at surface structure.

10.2.3 Variables bound by Quantified NPs and Weak Crossover

We have seen above that natural languages have the equivalent to **variables**, namely, pronouns that are bound by a quantifier. Of course, if you accept traces, then **traces** are variables as well. One standard way to express binding of pronouns is to assume that the quantified NP and the pronoun bear the same index, and that pronouns are interpreted as variables, just like traces. Example:

(19)Every girl$_1$ liked a boy that liked her$_1$
a. SS: [$_S$ [$_{NP,1}$ every girl] [$_{VP}$ liked [$_{NP,2}$ a boy that liked her$_1$]]]
b. LF: [$_S$ [$_{NP,1}$ every girl] [$_S$ e$_1$ [$_{VP}$ liked [$_{NP,2}$ a boy that liked her$_1$]]]]

We should assume a syntactic rule and semantic interpretation for relative clauses that, together with the other syntactic and semantic rules, will yield the following interpretation, given here in predicate logic. Note, in particular, that *her* is interpreted as the variable related to the girl.

(20)For every girl x there is a boy y that x liked, and y liked x, too.
x[girl(x) → y[boy(y) liked(x, y) liked(y, x)]]

One interesting restriction for quantified NPs and pronouns is that the quantified NP has to c-command any variable (pronoun or empty element) that it binds. For example we could have derived another logical form in which the indefinite NP has wide scope. However, our sentence doesn't have the corresponding reading. That is, the indefinite NP *a boy that likes her* always has narrow scope with respect to *every girl*, if *every girl* binds *her*. The reason why the second LF is not available is that the pronoun must be c-commanded by *every girl* in order to be bound by it, which rules out the corresponding logical form:

(21)Faulty LF: [$_S$ [$_{NP,2}$ a boy that liked her$_1$] [$_S$ [$_{NP,1}$ every girl] [$_{VP}$ liked e$_2$]]]

This prepares us to discuss a particularly striking similarity between overt syntactical movement and available readings. It concerns cases which are known as **weak crossover**. What is weak crossover? First, observe that we can form constituent questions from NPs in object position:

(22)a. Which boy$_1$ does the girl miss e$_1$?
b. Which boy does the woman like e$_1$?

Second, the following coindexations are possible, even though the pronoun precedes its “antecedent” (sometimes that is called **cataphora**, in contrast to **anaphora**):

(23)a. The girl that he$_1$ likes misses the boy$_1$.
b. His$_1$ mother likes the boy$_1$

Now, it turns out that the object NPs cannot be questioned in these sentences:

(24)a. *Which boy$_1$ does the girl that he$_1$ likes miss e$_1$?
b. *Which boy$_1$ does his$_1$ mother like e$_1$?

Why is this so? The normal conditions of binding theory don't exclude structures like (24.a,b). Notice that wh-words can bind pronouns, as shown in the following example:

(25) Which boy$_1$ did the girl ask e$_1$ to recite his$_1$ favourite poem?

Let us here just state the following generalization: a moved phrase (like a wh-pronoun) cannot bind **first** a pronoun and **then** a trace, for whatever reason (this is called the "leftness condition"). The interesting thing is that we find similar restrictions when we look at comparable structures on Logical Form:

(26)a. The girl that he likes misses every boy.
b. His mother likes every boy.

These examples cannot have the interpretations given by the following LF's:

(27)a. *[every boy$_1$ [the girl that he$_1$ likes misses e$_1$]]
'For every boy x, the girl that likes x misses x'
b. *[every boy$_1$ [his$_1$ mother likes e$_1$]]
'For every boy x, x's mother likes x'

So, whatever explains weak crossover for WH movement, should explain the unavailability of the readings for quantifiers. In particular, if weak crossover for WH movement is explained by syntactic rules, then the same syntactic rules should apply to the LF's of the QR structures.

10.3 Varieties of Quantifier Raising

10.3.1 Quantifier Raising to VP

In this section we will discuss evidence that object quantifiers need not raise to a position adjoining a sentence. Consider the following example, which involves VP ellipsis:

(28)A member of the Expose the Right group attended every public event held in New Hampshire, and a member of the Rainbow Coalition did, too.
(i) There is a member of the ER group that attended every public event, and there is a member of the RC that attended every public event.
(ii) For every public event there was a member of the ER group that attended it, and for every public event there was a member of the RC that attended it.

We find two readings: The object quantifier *every public event held in NH* has either narrow scope in both clauses (i), or wide scope in both clauses (ii). Notice that we do not find "mixed" readings. That is, (28) cannot be understood in either one of the following ways:

(28)(iii) There is a member of the ER group that attended every public event, and for every public event there was a member of the RC that attended it.

(iv) For every public event there was a member of the ER group that attended it, and there is a member of the RC that attended every public event.

We expect, of course, that our theory will predict the absence of such readings.

We can deal with the wide-scope reading (ii) as follows:

(29)$[_S$ [*every public event*] λ2[[*a member of ER*] λ1[$_S$ e_1 $[_{I'}$ PAST $[_{VP}$ *attended* e_2]]] *and*
[*a member of RC*] λ1[[$_S$ e_1 $[_{I'}$*did* $\underline{[_{VP}\ \textit{attended}\ e_2}$]]]]]

I have underlined the part that is the result of spell-out of VP-ellipsis. Notice that the antecedent is the VP [*attended* e_2] here. This predicts that the object position e_2 is bound by the same quantifier, which is in object position of the first clause but by quantifier raising gets scoped over the whole sentence. (That's a remarkable success story for the quantifier raising account!).

What about the narrow-scope interpretation of the object NP? Obviously, the quantifier should stay within the VP so that it can be part of the spell-out of VP ellipsis. This is suggested by the following paraphrase of (i):

(30)A member of the Expose the Right group attended every public event held in New Hampshire, and a member of the Rainbow Coalition <u>attended every public event held in New Hampshire</u>.

But if object quantifiers have to undergo quantifier raising, and quantifier raising is to a position adjoined to the sentence, and VP-ellipsis just repeats the material within the VP, then we obviously have a problem here.

One solution is to assume that quantifiers can be raised to adjoin to the level of VP, as an option:

(31)$[_S$ [*a member of ER*] λ1[$_S$ e_1 $[_{I'}$ PAST $[_{VP}$ *every public event*] λ2$[_{VP}$ *attend* e_2]]]] *and*
$[_S$ [*a member of RC*] λ1[$_S$ e_1 $[_{I'}$ *did* $\underline{[_{VP}\ \textit{every public event}]\ \lambda 2[_{VP}\ \textit{attend}\ e_2}$]]]]

But now we have to say how a construction of a quantifier like *every public event* adjoined to a VP with a trace like λ2$[_{VP}$ *attended* e_2] means. Note that we have a type mismatch here; the quantified NP is of type (et)t, and the VP that contains a trace is of type eet. In particular, the transitive verb is of type eet, application to the meaning of the empty element e_2 results in et, and lambda-abstraction again results in type eet.

Two strategies are possible here. First, we can give an interpretation rule for quantifiers when applied to expressions of type eet, as follows:

(32)If α is of type (et)t and β is of type eet, then $[\![[\alpha\ \beta]]\!] = \lambda x[[\![\alpha]\!](\lambda y[[\![\beta]\!](y)(x)])]$

The idea here is to temporarily bind the subject argument x and the object argument y by a variable, which results in a meaning of type t; then we abstract over the object argument y, which results in a meaning of type et; then we apply the quantifier to it; and finally we abstract over the subject argument x which is to be filled in the next step.

The other strategy is to do all that in syntax. We can assume that the VP contains the subject argument; we did this in section (3) when we discussed X-bar theory. But the expression that fills the subject argument always has to be moved in surface structure, which presumably is interpreted in the same way as quantifier raising:

(33)a. Underlying structure:
$[_{S}$ $[_{I'}$ PAST $[_{VP}$ [*a member of ER*] $[_{V'}$ *attend* [*every public event*]]]]

b. Surface structure:
$[_{S}$ [*a member of ER*] λ1[$[_{I'}$ PAST $[_{VP}$ e_1 $[_{V'}$ *attend* [*every public event*]]]]

c. Logical Form:
(i) $[_{S}$ [*a member of ER*] λ1$[_{I'}$ PAST [*every public event*] λ2$[_{VP}$ e_1 $[_{V'}$ *attend* e_2]]]
(ii) $[_{S}$ [*every public event*] λ2$[_{S}$ [*a member of ER*] λ1$[_{I'}$ PAST $[_{VP}$ e_1 $[_{V}$ *attend* e_2]]]]

Notice that the expressions that underwent quantifier raising adjoin to an expression that is of type et in every case.

Both ways of analyzing our example predict that there are no "mixed" readings. Either the quantified NP *every public event* scopes over the whole sentence, generating the wide-scope interpretation, or it stays within the VP, generating the narrow-scope reading.

10.3.2 Why do Quantifiers Raise?

A natural question is why quantifiers undergo raising on Logical Form. A natural answer is because otherwise they could not be interpreted. This is evident with object quantifiers. We have analyzed quantified NPs as being of type (et)t, and transitive verbal predicates as being of type eet. When we assume that NPs have to be interpreted as **arguments** of verbal predicates, we have a type mismatch here. Quantifier raising can be seen as a way to solve this type mismatch. In essence, quantifier raising consists in filling the argument position with a variable of the proper type (e), and binding this variable later by the quantifier.

This type of reasoning does not hold for subject quantifiers. They are of type (et)t, verbal predicates are of type et, and so the subject quantifier can be applied to the verbal predicate directly. But we have seen that subject quantifiers can undergo raising as well. A subject quantifier can have scope over an object quantifier; this is even the more natural interpretation in many cases.

The analysis illustrated with example (33) answers this concern as follows: Subject NPs have to raise for independent reasons. They have to end up in a position in which they c-command the I′ constituents for purely syntactic reasons, so they cannot be interpreted in situ.

10.3.3 Quantifiers within NP

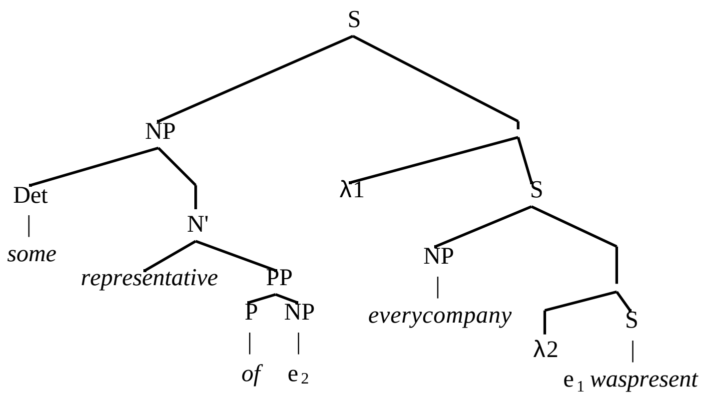
So far we have considered cases of scope ambiguity in which both quantifiers where co-arguments of a predicate. Now consider the following example, where one quantifier is part of another quantifier:

(34)Some representative of every soft drink company was present.
(i) There is a person x such that for every company y, x represents y, and x was present.
x[person(x) y[company(y) → represents(x, y)] present(x)]
(ii) For every company y there is a person x who represents y, and x was present.
y[company(y) → x[person(x) represents(x, y) present(x)]]

We have the two indicated readings. The wide-scope intepretation of the embedded quantifier *every company* (i.e., reading ii) is even more prominent than the other reading.

First, let us consider reading (i). It should be easy to derive, as it corresponds to the surface order of the quantifiers. However, we have a problem if we assume that quantifiers have to raise to the level of sentences, or to the level of an expression of type t. For the only structure that we get is the following:

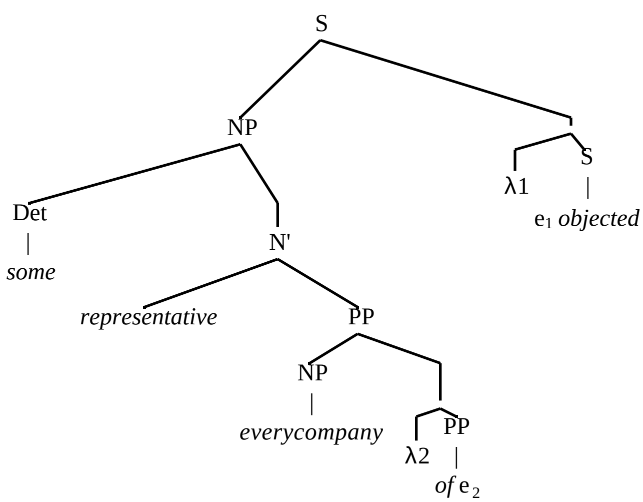
(35)



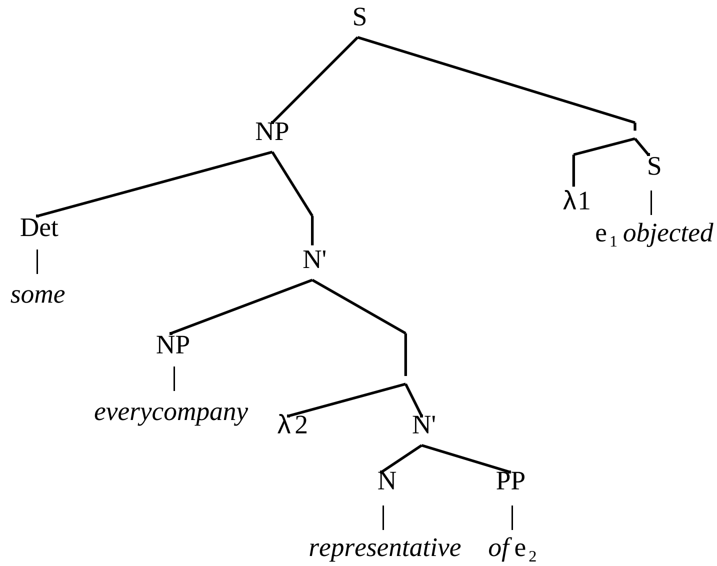
Here, the embedded NP *every company* has been raised first, and then the rest of the NP has been raised. The problem with this structure is that the NP *every company* is not in a position to bind its trace e_2, as it does not c-command that trace.

So we have to assume that *some company* is interpreted “within” its host NP. There are options like the following ones:

(36)



(37)



The problem with (36) is that we haven’t interpreted PP’s as expressions of type t. They are either expressions of type (et)et that modify a noun, or perhaps (in the case of PPs to relational nouns like *representative*) of type e. Hence the type of the epression [λ2 PP] is e(et)et or ee, certainly not what a quantifier like *a company*, which is of type (et)t, expects. A similar problem arises with (37): N′ is of type et, the constituent [λ2 N′] is then of type eet, again not what a quantifier expects.

There are two ways to proceed at this point: Either we allow that quantifiers come in other types as well (a possibility that is explored later, for some cases). Or we find a way to make PPs or N′s of the required type, namely, t. This is perhaps not such an implausible proposal, given the fact that we can paraphrase reading (i) of (34) as follows, even though the result sounds a bit clumsy:

(38)a. Some representative that was of every soft-drink company objected.
b. Someone who is a representative of every soft-drink company objected.

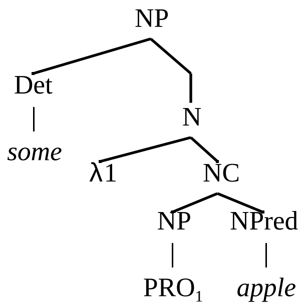
Let us see how this type of solution can be worked out with respect to the structure (37). We have to assume here that the noun *representative* actually contains an expression of type t, something like ‘x is a representative’, and that we lambda-abstract over x before it is combined with the determiner *a*, as in λx[x is a representative].

Which type of element is x? So far we have just one type of empty element, for which we used the letter e with indices; this type was created by movement in surface structure or logical form. But in the current case there is no movement. We find empty elements without movement in other cases, too, most prominently in infinitival constructions. They are typically indicated by PRO. In the following example, this PRO is bound by the subject, *Molly*:

(39)Molly$_1$ tried [PRO$_1$ to wake Leopold up].
‘Molly tried the following: Molly wakes up Leopold.’

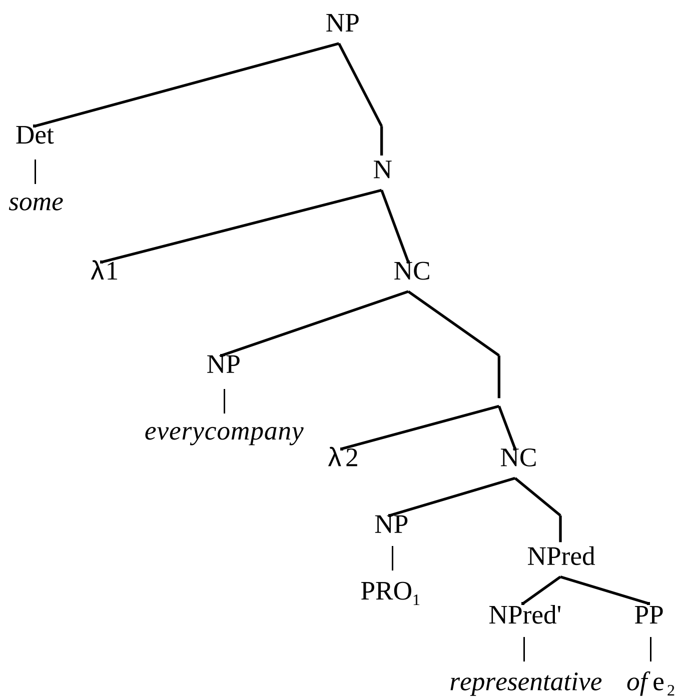
Let us then assume that the variable in [x is a representative] is a PRO as well. We then analyze an NP like *an apple* as follows:

(40)



Here, NC stands for “nominal clause”; it is an expression of type t. It consists of a PRO-NP as a subject, and of a nominal predicate NPred. The empty subject is made a function by the lambda term. The type of N is then, as before, et. The new category NC provides a landing site for quantifiers originating from whithin NPred:

(41)



This gives us the intepretation we are after, and we can assume throughout that quantifiers are of type (et)t.

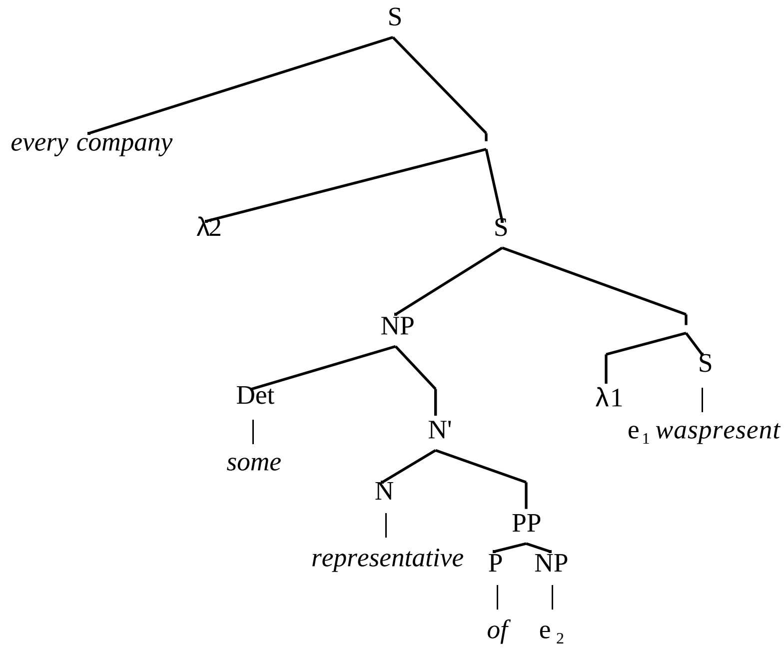
10.3.4 Inverse Linking

Let us now take up the second interpretation of (34), repeated here:

(42)Some representative of every soft drink company was present.
(ii) For every company y there is a person x who represents y, and x was present

We can deal with such cases if we assume that quantifiers can move out of other quantifiers (at least when they are within a PP. In contrast, a sentence constitutes an island, cf. the non-availability of the wide-scope reading for *a person that represented every company was present*).

(43)



But there is a problem with this solution: NPs are syntactic islands for regular movement operations.

(44)*Every company$_1$ she talked to [some representative of e$_1$] expected lower earnings.

Notice, in contrast, that the movement postulated in (41) is fine. In the following sentence, *a representative*... occurs in a predicative position, presumably a NPred.

(45)[every company$_1$ she was a representative of e$_1$] expected lower earnings.

The structure (43) is problematic for purely semantic reasons as well (cf. Heim & Kratzer 1998). Consider the following example:

(46)Neither a representative of every company nor John was there.

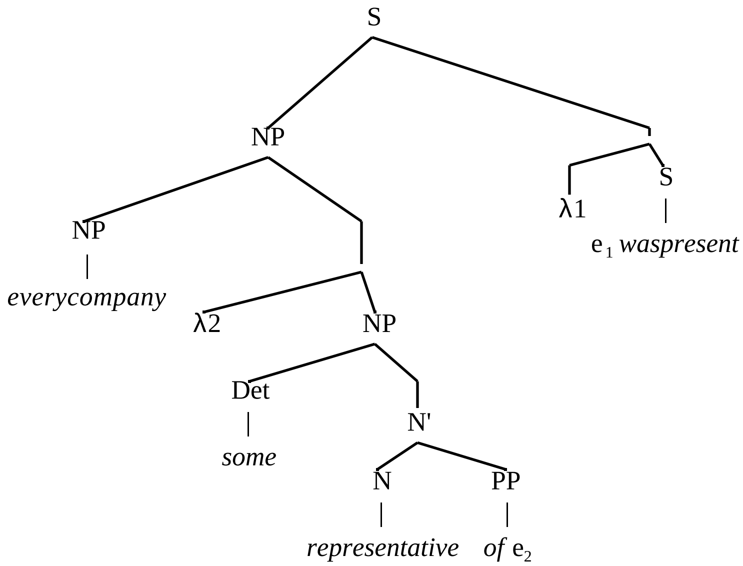
Under the wide-scope interpretation, this sentence says that John wasn't there, and that there is at least one company of which no representative was there either. But see what we get if *every company* adjoines to the sentence:

(47)[every company] λ1[[neither [a representative of e$_1$] nor John] was there]

This will be interpreted as: For EVERY company x it holds that neither a representative of x nor John was there. This is too strong; (46) can be true already if SOME company didn't send a representative.

So we might have to assume that quantifiers can move to adjoin to a NP, after all:

(48)



But this solution, of course, requires that the quantifier *every company* can adjoin to the structure [λ2 NP], which presumably is of type e((et)t) — not what a quantifier of type (et)t would expect! We will take up this issue when we talk about flexible quantifier types, in section 10.5 below.

10.4 Quantifier Storage

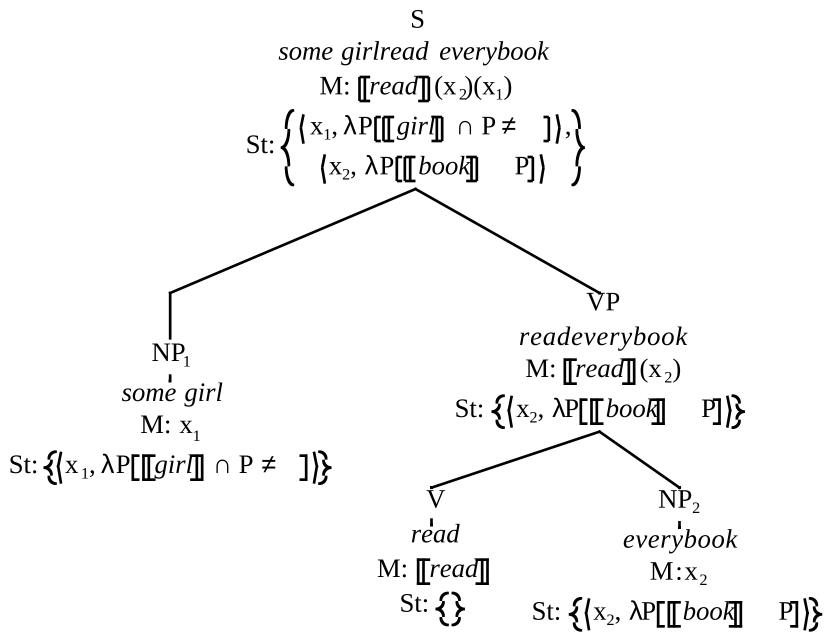
So far we have assumed that quantifier raising creates a distinct syntactic structure, Logical Form, that is the input to semantic interpretation. There are possible objections against this procedure. In particular, it appears as if we first transform the syntactic structure that we see or hear (the surface structure) in something that may be quite different, which we then send to the semantic component of the grammar. It is, in a sense, as if we would first translate English into another language, one that is more managable for semantic interpretation. Granted that it is not just any other language, but one that stands in well-defined syntactic relation to the surface structure. But everything else being equal, one can make the case that it would be more attractive if we could interpret surface structures directly.

There is a way to treat object quantifiers and scope ambiguities wholly within semantic interpretation, in a way that mimicks LF movement in semantics. This proposal goes back to Robin Cooper's 1977 dissertation (see also Cooper 1983, *Quantification and Syntactic Theory*.) and is called **Quantifier Storage** or **Cooper Storage**.

The basic ideas of quantifier storage is the following: The input for semantic interpretation is surface structure, there is no need for a level of logical form. But the semantic representations have to be a bit more complex. In addition to the ordinary semantic representations, they contain a **storage** device that can “store” quantifiers and other scope-bearing elements for a later application.

More precisely, a quantified NP is interpreted in the following way. First, as before, every NP gets a unique index. Then, at the position in the syntactic tree where the quantified NP is situated, the NP is interpreted as an indexed variable. This is of course similar to the way we interpret traces in the LF movement account. But notice that the variable is a semantic element, not a syntactic constituent. At the same time, the quantifier, together with the variable, is put into storage. In the course of the derivation we keep track of the storage; it may even increase because other quantifiers are added to it. Finally, at certain positions in the derivation of the meaning of the syntactic structure, the stored quantifier may be retrieved from the store, and applied. In the following example of a derivation; “M:” gives the meaning and “St” the store; the store is given as a set, with elements that are pairs of the indicated variable and the meaning of the NP as a quantifier.

(49)



We arrive at a meaning (under the top S node) that expresses a simple proposition, $[\![\textit{read}]\!](x_2)(x_1)$, and has a store with two elements that originated from the quantified NPs and were passed along in the derivation from the bottom to the top. We are not quite done yet, because the store has to be emptied. The rule for emptying the store is simple: Apply the quantifier to the lambda-abstract of the meaning, using the corresponding variable. The order in which the store is emptied is not fixed (notice that the store is represented as a set, and the order in which we list the elements of a set is irrelevant!). The two possible ways of emptying the store in the case at hand lead to the following results:

(50) Emptying the store, $x_2, \lambda P[[\![\textit{book}]\!] \quad P]$ first, then $x_1, \lambda P[[\![\textit{girl}]\!] \cap P \neq \quad]$:

a. $\lambda P[[\![\textit{book}]\!] \quad P](\lambda x_2[[\![\textit{read}]\!](x_2)(x_1)])$
   $= [[\![\textit{book}]\!] \quad \lambda x_2[[\![\textit{read}]\!](x_2)(x_1)]$

b. $\lambda P[[\![\textit{girl}]\!] \cap P \neq \quad](\lambda x_1[[\![\textit{book}]\!] \quad \lambda x_2[[\![\textit{read}]\!](x_2)(x_1)])$
   $= [\![\textit{girl}]\!] \cap \lambda x_1[[\![\textit{book}]\!] \quad \lambda x_2[[\![\textit{read}]\!](x_2)(x_1)] \neq$

(51) Emptying the store, $x_1, \lambda P[[\![\textit{girl}]\!] \cap P \neq \quad]$ first, then $x_2, \lambda P[[\![\textit{book}]\!] \quad P]$:

a. $\lambda P[[\![\textit{girl}]\!] \cap P \neq \quad](\lambda x_1[[\![\textit{read}]\!](x_2)(x_1)])$
   $= [\![\textit{girl}]\!] \cap \lambda x_1[[\![\textit{read}]\!](x_2)(x_1)]$

b. $\lambda P[[\![\textit{book}]\!] \quad P](\lambda x_2[\ [\![\textit{girl}]\!] \cap \lambda x_1[[\![\textit{read}]\!](x_2)(x_1)])$
   $= [\![\textit{book}]\!] \quad \lambda x_2[\ [\![\textit{girl}]\!] \cap \lambda x_1[[\![\textit{read}]\!](x_2)(x_1)]$

We see that the two possible ways of emptying the store gives us the two possible readings of the sentence.

We have seen that there are syntactic restrictions for the availability of wide-scope interpretations of quantifiers. These syntactic restrictions can be rephrased in semantic terms, and they may even look more convincing there. For example, we can assume that the store has to be emptied as soon as, in semantics, the type t (which corresponds to sentences) is reached. This immediately predicts that a quantifier that occurs within a relative clause cannot have scope over the matrix sentence (the sentence that embeds the relative clause).

The intepretation rules for a grammar with quantifier storage are naturally more complex. First of all, we have to make sure that the store of a complex expression is the sum of the stores of its parts. Let me write $[\![\alpha]\!]_S$ for the store of the expression α, then we have the following rule:

(52)a. $[\![[\alpha\ \beta]]\!] = [\![\alpha]\!]([\![\beta]\!])$ or $[\![\beta]\!]([\![\alpha]\!])$, depending which is well-formed;
b. $[\![[\alpha\ \beta]]\!]_S = [\![\alpha]\!]_S \quad [\![\beta]\!]_S$

These rules show how the content of the store is projected from smaller expressions to larger expressions. How do items get into the store? The following rule for indexed NPs illustrates that:

(53)a. $[\![[_{NP}\ \alpha]_i]\!] = x_i$
b. $[\![[_{NP}\ \alpha]_i]\!]_S = [\![\alpha]\!]_S \quad \{\ x_i, [\![\alpha]\!]\ \}$

That is, an indexed NP with index i is interpreted as a variable x_i, and the NP contributes the pair consisting of the variable and the meaning of α to the store. If the meaning of α contained a store already (as in *a representative of every company*), it is simply combined with this element.

Non-indexed simple expressions have an empty store, of course.

We are not quite done yet, because so far we have generated only interpretations that consist of a "meaning" and a "store". We can derive the regular interpretations of a sentence α as follows (notice that this is a set, because, in principle, α may be ambiguous!)

First, we define the set of **reduced meanings** $[\![\alpha]\!]_R$ of an expression (a sentence) α. This is an auxiliary notion, defined recursively in the following way:

(54)a. Basic case: If $[\![[_S\ \alpha]]\!] = M$ and $[\![[_S\ \alpha]]\!]_S = S$, then $M, S \quad [\![[_S\ \alpha]]\!]_R$.

b. Recursive clause: If $M, S \quad [\![[_S\ \alpha]]\!]_R$ and $x_i, Q \quad S$,
then $Q(\lambda x_i\ [M]), S$ ¾ $\{\ x_i, Q\ \} \quad [\![[_S\ \alpha]]\!]_R$.

Now we can define the notion of the set of **interpretations** of a sentence α, for which we write $[\![\alpha]\!]$, as before. It is the set of reduced meanings that are maximally reduced, that is, whose store is empty:

(55) $[\![[_S\ \alpha]]\!] = \{M|\ M, \qquad [\![[_S\ \alpha]]\!]_R\}$

10.5 Object Quantifiers and Flexible Types

10.5.1 Interpretation of Object Quantifiers in situ

Quantifier Raising or Quantifier Storage are not the only ways to deal with object quantifiers. Another strategy is the one we applied for Boolean conjunction and disjunction, which can exist in different types. We can assume the same for quantifiers.

Object quantifiers can be assigned the type (eet)et; they take a transitive verb meaning (type eet) and give an intransitive verb meaning (type et). We get the following interpretation for *every book* in subject position (for which I write NPs) and in object position (for which I write NPo). I disregard variable assignments here.

(56)a. $⟦[_{NPs}\ [_{Dets}\ \textit{every}]\ [_N\ \textit{book}]]⟧ = \lambda P \quad D_{et}[⟦\textit{book}⟧ \quad P]$
b. $⟦[_{NPo}\ [_{Deto}\ \textit{every}]\ [_N\ \textit{book}]]⟧ = \lambda R \quad D_{eet}\lambda x \quad D_e[⟦\textit{book}⟧ \quad \lambda y[R(y)(x)]]$

In (b), R stands for the meaning of the transitive verb, and x is the subject argument that will be filled later. The difference in interpretation is can be traced back to a difference in the determiner meaning:

(57)a. $⟦[_{Dets}\ \textit{every}]⟧ = \lambda P' \quad D_{et}\lambda P \quad D_{et}[P' \quad P]$
b. $⟦[_{Deto}\ \textit{every}]⟧ = \lambda P' \quad D_{et}\lambda R \quad D_{eet}\lambda x \quad D_e[P' \quad \lambda y[R(y)(x)]]$

This allows for representations like the following; I specify here the type of the expressions, and I give the derivation in form of a tree that indicates the syntactic structure. The semantic operation is always functional application.

(58)

somegirlreadeverybook, t
$[⟦\textit{girl}⟧ \cap \lambda x[⟦\textit{book}⟧ \quad \lambda y[⟦\textit{read}⟧(y)(x)] \neq \quad]$

somegirl, (et)t
$\lambda P[⟦\textit{girl}⟧ \cap P \neq \quad]$

readeverybook, et
$\lambda x[⟦\textit{book}⟧ \quad \lambda y[⟦\textit{read}⟧(y)(x)]$

read, eet
$⟦\textit{read}⟧$

everybook, (eet)et
$\lambda R\lambda x[⟦\textit{book}⟧ \quad \lambda y R(y)(x)]$

The type of ambiguity we have assumed for object NPs and for subject NPs (or rather, object determiners and subject determiners) is systematic, not like the ambiguity of words like *pen.* In particular, whenever we have the meaning of a subject quantifier Q_S, we can derive the meaning of the corresponding object quantifier Q_O in the following way:

(59) $Q_O = \lambda R\lambda x[Q_S(\lambda y R(y)(x))]$

This procedure can be easily generalized for quantifiers of n-place predicates in general: They take an n-place predicate and yield a (n-1)-place predicate. Hence they should be of the following type:

(60) Quantifier for n-place predicate: $(\tau_n)\tau_{n-1}$, where τ_k: type of a k-place predicate.

The case of subject quantifiers, type (et)t, turns out to be a particularly simple subcase: It is a quantifier that changes a 1-place predicate to a 0-place predicate. Of course, determiners will then have different types as well. Determiners take a noun (type et) and give a quantifier, hence are of the following type:

(61) Determiners for n-place predicate: $(et)(\tau_n)\tau_{n-1}$

We say that the type of NPs (and of determiners, etc.) is **flexible**. That is, these expressions are not assigned to a unique type, but to a whole family of types. We can choose the type that fits to a given construction. Emmon Bach called this **shake'n'bake semantics**: We take an expression, "shake" it so that it gets a suitable type, and then "bake" it together with another expression.

10.5.2 Scope Ambiguity and Type Flexibility

Our next question should be: Is it possible to account for scope ambiguities within a framework that uses flexible types for quantifiers?

We have analyzed transitive verbs as being of type eet, object quantifiers as being of type (eet)et, and subject quantifiers as being of type (et)t. We have seen that this allows for derivations in which the subject quantifier takes scope over the object quantifier.

Is there a derivation in which the object quantifier can scope over the subject quantifier? One possibility would be to allow for a combination of a subject NP and a transitive verb. We would have to assume subject and object NPs like the following, in addition to the ones in (56):

(62)a. $[\![[_{NPs}\ [_{Dets}\ \textit{some}]\ [_N\ \textit{girl}]]]\!] = \lambda R\ \ D_{eet}\lambda y\ \ D_e[[\![\textit{girl}]\!] \cap \lambda x[R(y)(x)] \neq\ \ ]$

b. $[\![[_{NPo}\ [_{Deto}\ \textit{every}]\ [_N\ \textit{book}]]]\!] = \lambda P\ \ D_{et}[[\![\textit{book}]\!]\ \ \ P]$

This grammar then would endorse derivations like the following:

(63)

somegirllikedeverybook, t
$[[\![\textit{book}]\!]\ \ \lambda y[[\![\textit{girl}]\!] \cap \lambda x[[\![\textit{liked}]\!](y)(x)] \neq\ \ ]]$

somegirl liked, (et) — $\lambda y[[\![\textit{girl}]\!] \cap \lambda x[[\![\textit{liked}]\!](y)(x)] \neq\ \ ]$

everybook, (et)t — $\lambda P[[\![\textit{book}]\!]\ \ \ P]$

somegirl, (eet)et — $\lambda R\lambda y[[\![\textit{girl}]\!] \cap \lambda x[R(y)(x)] \neq\ \ ]$

liked, eet — $[\![\textit{liked}]\!]$

However, the resulting syntactic structure runs against well-established assumptions for the constituent structure of English, which indicate that the verb forms a constituent with the object. This even holds in cases in which the object has wide scope. Reconsider our example (28), repeated here:

(64)A member of the Expose the Right group attended every public event held
in New Hampshire, and a member of the Rainbow Coalition did, too.

This has a reading in which the object NP has wide scope: 'For every public event held in New Hampshire x, there was a member of the Expose the Right group and a member of the Rainbow Coalition that attended x'. Obviously, a structure like the one in (63) would not give us that result.

There is another way to achieve an in-situ interpretation of object quantifiers. We can assume that verbs can take quantifiers (type (et)t) directly as **arguments**. For example, an intransitive verb is not only of type et, but may also be of type ((et)t)t. Consider the following two versions of the meaning of *sleep*:

(65)a. Ordinary meaning: $[\![\textit{sleep}]\!] = \lambda x\ \ D_e[\text{x sleeps}]$

b. Type-lifted meaning: ⟦*sleep′* ⟧ = λQ $D_{(et)t}$[Q(⟦*sleep*⟧)]

Obviously, these two meanings would give us the same result in sentences like *Every girl sleeps*. However, we can also assume meanings for object NPs that don't give us ordinary predicates of type et as result, but predicates of type ((et)t)t. These object NPs then must be of type (eet)((et)t)t.

(66)a. Ordinary meaning: ⟦*every book*⟧ = λR D_{eet}λx D_{e}[⟦*book*⟧ λy[R(y)(x)]]
b. Type-lifted meaning: ⟦*every book'* ⟧ = λR D_{eet}λQ $D_{(et)t}$[⟦*book*⟧ λy[Q(R(y))]]

With this we can have derivations like the following:

(67)

some girl liked every book, t
⟦⟦*book*⟧ λy[⟦⟦*girl*⟧ ∩ ⟦*like*⟧(y)] ≠]⟧

some girl, (et)t
λP[⟦⟦*girl*⟧ ∩ P ≠]

liked every book, ((et)t)t
λQλx[⟦⟦*book*⟧ λy[Q(⟦*like*⟧(y))]]⟧

liked, eet
⟦*liked*⟧

every book, (eet)((et)t)t
λRλQ[⟦⟦*book*⟧ λy[Q(R(y))]]⟧

Notice that the VP, *liked every book*, has a meaning that forces any subject quantifier into a position within the scope of the object quantifier.

10.5.3 Quantifiers in Quantifiers

Let us now consider quantificational NPs within quantificational NPs, which we have discussed with the following example:

(68)Some representative of every company was present.

We have seen in sections 10.3.3 and 10.3.4 that we can handle the two readings of this sentence in the LF-movement account. However, we also have seen that we probably have to give up the idea that within this account we can work with just one type for quantifiers, (et)t. The wide-scope reading of *every company* seems to require that this quantifier is attached to its host quantifier, *some representative of...*, which is not of the proper type, et.

The question arises whether we can deal with the readings of (68) within the flexible-type approach. It turns out that we already have the tools for the narrow-scope interpretation of *every company*. If we assume that *representative* is of type eet, then we have to apply a quantifier meaning of the type of (56.b).

(69)a. ⟦[$_{NP}$ [$_{Det}$ *some*] [$_{N'}$ *representative* [$_{PP}$ *of* [$_{NP}$ *every company*]]]]⟧
b. = ⟦*some*⟧(⟦*every company*⟧(⟦*representative*⟧))
c. = ⟦*some*⟧(λR D_{eet}λx D_{e}[⟦*company*⟧ λy[R(y)(x)]](⟦*representative*⟧))
d. = ⟦*some*⟧(λx D_{e}[⟦*company*⟧ λy[⟦*representative*⟧(y)(x)])
e. = λP′ D_{et}λP D_{et}[P′ ∩ P ≠](λx D_{e}[⟦*company*⟧ λy[⟦*representative*⟧(y)(x)]))
f. = λP D_{et}[λx D_{e}[⟦*company*⟧ λy[*representative*⟧(y)(x)] ∩ P ≠]

Combining this with a VP meaning like ⟦*was present*⟧ we get:

g. $\lambda x \quad D_e[⟦company⟧ \quad \lambda y[⟦representative⟧(y)(x)] \cap ⟦be\ present⟧ \neq \quad]$

A moment's thought will show you that this is the right meaning. It says that there is at least one x that is present and for which it holds that the set of things y that x represents includes all companies.

What about the wide-scope interpretation? To derive this we have to assume that both the determiner *some* and the the determiner *every* are able to combine with other types than the ones considered so far. As for *some*, we have to allow for the following interpretation:

(70) $⟦some⟧ = \lambda R \quad D_{eet}\lambda x \quad D_e\lambda P \quad D_{et}[R(x) \cap P \neq \quad]$

This is a meaning of type (eet)e(et)t, not as the usual subject determiners which are of type (et)(et)t, nor like the object determiners which are of type (et)(eet)et. The idea is that *some*, in this function, does not take a regular nominal predicate of type et, but rather a relational noun meaning of type (eet). One of the arguments of this relational noun survives, and shows up as an extra e argument in the meaning of a construction like *some representative*:

(71) $⟦some⟧(⟦representative⟧) = \lambda x \quad D_e\lambda P \quad D_{et}[⟦representative⟧(x) \cap P \neq \quad]$

This meaning could now be combined with an expression of type e, as in the following example. (Recall that we assume that *of* is semantically vacuous).

(72) $⟦some⟧(⟦representative⟧)(⟦of⟧(⟦Coca\text{-}Cola⟧))$
$= ⟦some⟧(⟦representative⟧)(⟦Coca\text{-}Cola⟧)$
$= \lambda x \quad D_e\lambda P \quad D_{et}[⟦representative⟧(x) \cap P \neq \quad](⟦Coca\text{-}Cola⟧)$
$= \lambda P \quad D_{et}[⟦representative⟧(⟦Coca\text{-}Cola⟧) \cap P \neq \quad]$

So far this is just a way to parse the string a little bit differently, with the same semantic result. But we now can also assume a different interpretation of *every*:

(73) $⟦every⟧ = \lambda P \quad D_{et}\lambda S \quad D_{e(et)t}\lambda P' \quad D_{et}[P \quad \lambda x \quad D_e[S(x)(P')]]$

We then get the following intepretation:

(74) a. $⟦every⟧(⟦company⟧)$
$= \lambda S \quad D_{e(et)t}\lambda P' \quad D_{et}[⟦company⟧ \quad \lambda x \quad D_e[S(x)(P')]]$
b. $⟦every⟧(⟦company⟧)(⟦some⟧(⟦representative⟧))$
$= \lambda P' \quad D_{et}[⟦company⟧ \quad \lambda x \quad D_e[⟦representative⟧(x) \cap P' \neq \quad]]$

When we combine this further with a VP meaning like the one of *be present*, we finally arrive at the following:

(75) $⟦every⟧(⟦company⟧)(⟦some⟧(⟦representative⟧))(⟦be\ present⟧)$
$= [⟦company⟧ \quad \lambda x \quad D_e[⟦representative⟧(x) \cap ⟦be\ present⟧ \neq \quad]]$

That is, the set of companies is a subset of the entities x such that a representative of x was present. This is the right reading.

Of course, just stipulating that the new interpretations of *some* and *every* required for this derivation exist is not sufficient. We would now have to find the principles under which these new intepretations can be derived, and check whether they do not overgenerate in other cases. But this goes well beyond this introduction.