8 Pronouns and Variable Assignments

It is useful for this section to work with two slight changes of our previous notation:

First, the restriction of functions given in lambda terms can be expressed as in (c), not just as in (a) and (b).

- λx S[---], e.g. λx {x | x is a person}[x snores]
- λx[… | ---], e.g. λx[x is a person | x snores]
- λx … [---], e.g. λx x is a person [x snores]

Second, if $\Phi_1, \Phi_2, \ldots \Phi_n$ are truth values, then I will write $[\Phi_1, \Phi_2, \ldots \Phi_n]$ instead of MIN($\{\Phi_1, \Phi_2, \ldots \Phi_n\}$).

8.1 Pronouns as Placeholders

8.1.1 Pronouns, Antecedents, Variable Assignments

One important type of noun phrases that we have not considered so far are **pronouns**, as in the following examples:

(1) a. Leopold scratched **himself**.
b. Leopold loves **his** wife.
c. Leopold thinks that Molly loves **him**.
d. Leopold entered the house. **He** took off **his** boots.

In (1.a) we find a reflexive pronoun *himself.* It refers to Leopold. We say that *Leopold* is the **antecedent** of the pronoun *himself,* and we say that *himself* is a pronoun or, more generally, an **anaphoric expression**. In general, if a pronoun and its antecedent are arguments of the same verb, the pronoun comes in a reflexive form.

In (b) the pronoun is *his*. Again, *Leopold* is its antecedent; we could replace it by *Leopold's* without change of truth conditions. It occurs in determiner position, a so-called **possessive** pronoun. The phrase *his wife* means the same as the (stylistically awkward) *the wife of him*.

In (c) we have a case in which the pronoun occurs in an embedded sentence, with respect to its antecedent. And in (d) the pronoun occurs in a sentence that follows the sentence that contains its antecedent. For the time being we will disregard such cases, not so much because the pronouns need a different treatment but rather because the analysis of embedded sentences and of texts consisting of a sequence of sentences requires tools that we haven't yet developed.

It is customary to use **indices** to express the antecedent-pronoun relationship. Pronouns often lead to ambiguous structures, and this device allows us to disambiguate sentences for the sake of discussion:

(2) Molly introduced Leopold to Stephen. He obviously liked him.
a. Molly introduced Leopold$_1$ to Stephen$_2$. He$_1$ obviously liked him$_2$.
b. Molly introduced Leopold$_1$ to Stephen$_2$. He$_2$ obviously liked him$_1$.

What do pronouns mean? An assumption that is very plausible at first is that they are simply abbreviations of their antecedents:

(3) a. Leopold$_1$ scratched himself$_1$.
means: Leopold scratched Leopold.
b. Leopold$_1$ loves his$_1$ wife.
means: Leopold loves Leopold's wife.
c. Leopold$_1$ thinks that Molly loves him.
means: Leopold thinks that Molly loves Leopold.

This is certainly right if the antecedent is a name, like *Leopold*, or a definite NP, like *the stocky man.*. However, we will see later that this is not always the case. But for the time being we will concentrate on those cases for which a pronoun can be seen as a proxy for its antecedent.

8.1.2 Variable Assignments

How can we guarantee that a pronoun is interpreted as its antecedent? Notice that we cannot determine the interpretation of a pronoun by just looking at the pronoun itself. We rather must consider the **context** in which the pronoun occurs. In the context of (4.a) *himself* denotes something different than in the context of (4.b).

(4) a. Leopold$_1$ scratched himself$_1$.
b. Stephen$_1$ scratched himself$_1$.

We consider a pronoun a **placeholder** or a **variable** whose value has to be supplied from elsewhere, namely, from the antecedent.

We will have to design some mechanism for this supplying of a value. We will make use of the device of indexation that we introduced above. That is, we will assume that the syntactic structures, the input to the interpretation rules, come with indices, for which we will use natural numbers. The question now is: How should we interpret a variable like *himself*$_1$?

(5) $[\![himself_1]\!]$ = ?

This clearly depends on the **context** in which *himself*$_1$ is uttered. We should find a way in which this context-dependency can be formally expressed. The general solution to this problem is the notion of a **variable assignment**. A variable assignment is a rule that tells us how to assign a meaning to variables, in our case, how to assign a meaning to the index 1. In general, a variable assignment is a function from indices (variables) to meanings.

(6) Variable assignments: A function from indices (variables) to meanings.

In the case of (5) a variable assignment that does the job is the function { 1, LB }, which maps the index 1 to LB. We would achieve the same result with other assignments, like { 1, LB , 2, SD }, which is a variable assignment that assigns the index 2 to SD, which we need in case there is more than one antecedent-pronoun relationship:

(7) Leopold$_1$ thinks that Stephen$_2$ wants him$_1$ to be his$_2$ father.

I will use "g", "g′" etc. for variable assignments, and I will call the set of all variable assignments "G". I will render variable assignments, which are technically functions (sets of pairs) in a somewhat shorter way: Instead of "{ 1, LB , 2, SD }" I will write "[1 → LB, 2 → SD]".

Interpretations of expressions are dependent on a variable assignment. That is, they are functions from variable assignments to what we consider their usual meaning:

(8) ⟦*himself*$_1$⟧: a function from variable assignments G to D_e.

We can be more specific. The pronoun *himself*$_1$ is a function from variable assignments g that have the index 1 in their domain, and that map this index to a male person. Hence the meaning of *himself*$_1$, applied to that assignment, is the value of the assignment applied to the index 1:

(9) ⟦*himself*$_1$⟧ = λg 1 DOM(g), g(1) is male [g(1)],
= λg g(1) is male [g(1)]

This does not capture that the antecedent of *himself*$_1$ must be within its own clause, a condition that we happily leave to the syntactic component to check (namely, principle A of Binding Theory).[1]

If the meaning of pronouns is a function from variable assignments, then it will be handy to assume the same for meanings in general. After all, the syntactic distribution of pronouns is the similar to the syntactic distribution of **names**, and hence the semantic intepretation rules are made simple if they can expect the same type of meaning in each case:

(10)Leopold scratched {himself / Molly}.

But of course, a name is not dependent on any variable assignment. So we have meanings of the following sort:

(11)a. ⟦*Molly*⟧ = λg[MB] (= λg G[MB])
b. ⟦*snores*⟧ = λgλx[x snores]

(11.a) is a function that maps every variable assignment to Molly (later we will have to deal with the case on names with an index). And (b) is a function that maps every assignment to the function from entities x to truth values such that x is mapped to 1 if x sleeps, and else to 0. Technically these are functions from variable assignments, but the variable assignments somehow don't matter.

Our semantic composition rules will have to undergo certain changes as well. The most important composition rule is functional application. We used to write things like that:

(12)a. ⟦*snores*⟧(⟦*Molly*⟧)
b. = λx[x snores](MB)

This was possible as ⟦*snore*⟧ was a meaning of type et (a function from entities), and ⟦*Molly*⟧ a meaning of type e (an entity). But now it doesn't work anymore. If we use "a" as the type of assignments, then ⟦*snore*⟧ is of type aet, and ⟦*Molly*⟧ is of type ae, and these types do not fit:

(13)*λgλx[x snores](λg[MB])

But we can redefine functional application for meanings that are functions from assignments as follows:

(14)Functional application for meanings that are functions from assignments:
If ⟦α⟧ is of type aστ and ⟦β⟧ is of type aσ,
then ⟦α⟧(⟦β⟧) = λg g [DOM(⟦α⟧), g DOM(⟦β⟧) [⟦α⟧(g)(⟦β⟧(g))]

[1] An alternative for the treatment of reflexive pronouns is to analyze them as operators that reduce the arguments of verbs. For example, ⟦*herself*⟧ = λR D_{ee}λx x is female [R(x)(x)]. But this would not work for other pronouns, and we take here reflexives as the simplest pronouns with an antecedent in the same sentence.

That is, we get as result again a function from assignments g. Its domain will be the intersection of the domains of ⟦α⟧ and ⟦β⟧. Its value is what we get when applying ⟦α⟧ first to the assignment g and then to the result of ⟦β⟧ applied to g. To see how things work, consider first a case without pronoun:

(15)a. ⟦[$_S$ [$_{NP}$ *Molly*] [$_{VP}$ *snores*]]⟧
b. = λg g [DOM(⟦*Molly*⟧), g DOM(⟦*snores*⟧) [⟦*snores*⟧(g)(⟦*Molly*⟧(g))]
c. = λg g [DOM(⟦*Molly*⟧), g DOM(⟦*snores*⟧) [λg′λx[x snores](g)(λg″[MB](g))]
d. = λg G[λx[x snores](MB)]
e. = λg[MB snores]
f. = λg[1], if Molly snores,
= λg[0], if Molly doesn't snore.

I have used different variables for assignments to keep them apart as an aid to understand what's going on here. First, notice that the domain of both ⟦*Molly*⟧ and ⟦*snores*⟧ is the set of all variable assignments, G. Their intersection is G again, and this is the domain of the resulting meaning. As for it's value, it is a function from assignments g to ⟦*snores*⟧(g) applied to ⟦*Molly*⟧(g), which simply gives us the truth value of [MB snores], as the expressions in this sentence are not dependent on variable assignments.

Now, what about an expression that contains a pronoun?

(16)a. ⟦[$_S$ [$_{NP}$ *she*]$_1$ [$_{VP}$ *snores*]]⟧
b. = λg g DOM(⟦[$_{NP}$ *she*]$_1$⟧), g DOM(⟦*snores*⟧) [⟦*snores*⟧(g)(⟦[$_{NP}$*she*]$_1$⟧(g))]
c. with ⟦[$_{NP}$ *she*]$_1$⟧ = λg″ g″(1) is female}[g″(1)]:
= λg g(1) is female, g G
[λg′ Gλx D_e[x snores](g)(λg″ g(1) is female [g″(1)](g))]
d. = λg g(1) is female [λx D_e[x snores](g(1))]
e. = λg g(1) is female [g(1) snores]
f. = λg g(1) is female [1], if g(1) snores,
= λg g(1) is female [0], if g(1) doesn't snore.

We now get a function that is restricted to a certain subset of the assignment functions, namely to those that are defined for the index 1 and map this index to a female person. The function maps every assignment g that satisfies this condition to 1 if g(1) snores, and to 0 if g(1) doesn't snore. Hence this function is truly dependent on the assignment function g. This is as it should be. If we have two women, say, Molly and Millie, and Molly snores but Millie doesn't, then some of the functions in the domain of ⟦*she*$_1$ *snores*⟧ (the ones that map 1 to Molly) give us the value 1, and others (the ones that map 1 to Millie) give us the value 0.

This reflects the fact that, out of the blue, we don't know what the truth value of a sentence like *she snores* means unless we know what *she* is supposed to mean. The meaning of a sentence with a variable, like *she snores* is often called a **propositional function**, and contrasted with the meaning of a sentence that is not dependent on the intepretation of variables, which is called a **proposition**.

In this section, then, we have developed a way how to capture the fact that pronouns are dependent on the context in which they are uttered. We have done this by introducing the notion of a variable assignment, a a function from assignments to ordinary meanings. Variable assignments have no effect on expressions that do not contain pronouns, but they determine the meaning of pronouns. Of course, they then also determine the meaning of expressions that contain pronouns.

8.1.3 Supplying a Value

The next issue that we have to address now is how an antecedent can exert its influence on a pronoun that it binds. The antecedent should influence the choice of admissible variable assignments. Take the following example:

(17)Leopold$_1$ scratched himself$_1$.

The subject *Leopold*, with index 1, should restrict the assignments g to those that map the index 1 to Leopold. We can achieve this with the following interpretation:

(18)a. $[\![[_{NP}\ \textit{Leopold}]_1]\!]$
b. = λg g(1) = $[\![$Leopold$]\!]$(g) [g(1)]
c. = λg g(1) = λg′[LB](g) [g(1)]
d. = λg g(1) = LB [g(1)]

This is a function that is defined for all assignments g that map the index 1 to LB. It gives g(1) as a value, which is of course LB for all those functions.

Now we can interpret (17), as follows (this is a top-down derivation; you may also try it bottom-up):

(19)a. $[\![[_S\ [_{NP}\ \textit{Leopold}]_1\ [_{VP}\ [_V\ \textit{scratched}]\ [_{NP}\ \textit{himself}]_1]]]\!]$
b. = λg g DOM($[\![[_{VP}\ [_V\ \textit{scratched}]\ [_{NP}\ \textit{himself}]_1]]\!]$) g DOM($[\![[_{NP}\ \textit{Leopold}]_1]\!]$)
[$[\![[_{VP}\ [_V\ \textit{scratched}]\ [_{NP}\ \textit{himself}]_1]]\!]$(g)($[\![[_{NP}\ \textit{Leopold}]_1]\!]$(g))]
c. $[\![[_{VP}\ [_V\ \textit{scratched}]\ [_{NP}\ \textit{himself}]_1]]\!]$
d. = λg g [DOM($[\![\textit{scratched}]\!]$), g DOM($[\![[_{NP}\textit{himself}]_1]\!]$)
[($[\![\textit{scratched}]\!]$(g)($[\![[_{NP}\textit{himself}]_1]\!]$(g))]
e. = λg g(1) is male [λg′λxλy[y scratched x](g)(λg″[g″(1)](g))]
f. = λg g(1) is male [λxλy[y scratched x](g(1))]
g. = λg g(1) is male [λy[y scratched g(1)]}
h. $[\![[_{NP}\ \textit{Leopold}]_1]\!]$ = λg g(1) = LB [g(1)]
i. b. = λg g(1) is male, g(1) = LB [λg′λy[y scratched g′(1)](g)(λg″[g″(1)](g))]
k. = λg g(1) is male, g(1) = LB [λy D_e[y scratched g(1)](g(1))]]
l. = λg g(1) is male, g(1) = LB [g(1) scratched g(1)]
m. = λg g(1) is male, g(1) = LB [1], if g(1) scratched g(1),
= λg g(1) is male, g(1) = LB [0], if g(1) did not scratch g(1)

We get the following result: A function from assignments g that map g(1) to a male person and, specifically, to Leopold, and that give the value 1 if g(1) scratched g(1), and else 0.

This captures at least two important things. If Leopold doesn't exist, or if Leopold is not a man, then a sentence like *Leopold scratched himself* is weird. It is not simply false — the sentence *Leopold didn't scratch himself* is equally weird, and not true. This shows that the meaning components that Leopold exists and that Leopold is a man are **presuppositions** of the sentence.

But otherwise we are not quite there yet. We were quite happy to have a function from assignments to truth values for sentences with a pronoun that lacks an antecedent, like *She snores*. The truth value of this sentence depends indeed on the way how the pronoun *she* is interpreted. But for *Leopold*$_1$ *scratched himself*$_1$ things are different; the sentence expresses that the pronoun *himself* should be interpreted in the same way as *Leopold*.

How can we, then, arrive at a truth value? Intuitively, there must be at least one assignment for which the sentence is true. This condition has been called **existential closure**, and can be rendered as follows:

(20)a. A function f from a set of assignments G′ to D_t represents a true sentence if there is at least **one** g G′ such that f(g) = 1.

That is, under the principle of existential closure it is enough to find one assignment g such that the sentence meaning, when applied to g, gives us the value 1. For the sentence *Leopold*$_1$ *scratched himself*$_1$ to be true there must be one assignment for which the sentence is true; notice that every assignment for which it is true will give us the value for Leopold for *himself.*

8.2 Relative Clauses

Before we continue with the interpretation of quantifiers as antecedents of pronouns it is helpful to discuss a syntactic construction that we haven't analyzed so far, namely **relative clauses**.

8.2.1 Restrictive and Appositive Relative Clauses

There are two types of relative clauses, so-called **restrictive** relative clauses like (21) and so-called **appositive** (or non-restrictive) relative clauses like (22).

(21)a. Leopold watched the woman *who was talking*.
b. Stephen talked to a woman *who he had met in London.*

(22)a. Leopold watched the woman, *who was talking*.
b. Stephen talked to Molly, *who he had met in London.*

Both types of relative clauses are parts of noun phrases. The difference is that that restrictive relative clauses, more specifically, expand the noun of a noun phrase, whereas appositive relative clauses expand the noun phrase itself.

Consider (21.a). The relative clause apparently modifies the noun *woman* to form a nominal predicate with a more specific meaning, *woman who was talking*; to this we apply the meaning of the definite article. In a situation in which there are two women, one of whom is talking, the definite NP *the woman* would be inadequate because it would violate the uniqueness condition, but the NP *the woman who was talking* would be fine because the uniqueness condition is satisfied.

Now consider (22.a). Here the relative clause gives more information about a particular person, identified either by a name, *Molly,* or by a definite description, *the woman*. The sentence (with *the woman*) would be inadequate in a situation in which there is more than one woman, even if only one of them is talking.

There are certain formal differences between the two types of relative clauses. Relative clauses introduced by *that* are always restrictive (cf. *the woman that was talking*), and appositive relative clauses typically are separated by an intonational break, which is indicated in English orthography by a comma.

We then should assume the following syntactic rules:

(23)a. Restrictive relative clauses:N → N CP
b. Appositive relative clauses: NP → NP CP

Here I have used "CP", for **complementizer phrase**, for the category of relative clauses.

What are relative clauses, from a semantic viewpoint? In essence, they are like adjectives or preposition phrases. We essentially have the following meanings:

(24)a. ⟦*who was talking*⟧ = λx[x was talking]
b. ⟦*who he (= Stephen) met*⟧ = λx[Stephen met x]

In the case of appositive relative clauses, these predicates are applied to the entity that the NP refers to. This forms a separate predication in addition to the predication expressed by the main clause. We can paraphrase (22.a) as follows:

(25)Leopold watched Molly, who was talking.
'Leopold watched Molly, and the woman was talking.'

Notice that in the second conjunct, the meaning of the relative clause (= (24.a)) is applied to the meaning of the NP, ⟦*Molly*⟧. We should not be concerned here with the question how this meaning can be derived compositionally. But it is interesting to note that the content of the appositive relative clause is presupposed. For example, from *It is not the case that Molly, who was talking, watched Leopold* it still follows that Molly watched Leopold, which is typical for presuppositions.

In the case of restrictive relative clauses, the relative clause is combined with the meaning of a noun. Hence the type of relative clauses suggested in (24) wouldn't work; we cannot combine two meanings of type D_{et} by functional application. We have encountered a similar problem with adjectives, and suggested a way to derive the meaning of attributive (noun-modifiying) adjectives from predicative adjectives. The same technique works here as well:

(26) ⟦*who was talking*⟧ (as a restrictive relative clause): λPλx[P(x), x was talking]

This leads to derivations like the following:

(27)a. ⟦[$_N$ [$_N$ *woman*] [$_{CP}$ *who was talking*]]⟧
b. = ⟦[$_{CP}$ *who was talking*]]⟧(⟦[$_N$*woman*]]⟧)
c. = λPλx[P(x), x was talking](λx[x is a woman])
d. = λx[x is a woman, x was talking]

We get a function that maps an entity x to 1 iff it is both a woman and was talking, and else to 0. This is exactly what we want to have.

8.2.2 The Internal Structure of Relative Clauses

One crucial property of relative clauses is that one position within the clause is empty and corresponds to the relative pronoun. In (21.a) this is the subject, and in in (b), the object. In the following examples these empty positions are made visible, by the letter "e", and they are coindexed with the relative pronoun.

(28)a. who$_1$ [$_S$ [$_{NP}$ e]$_1$ [$_{VP}$ snored]]
b. who$_1$ [$_S$ [$_{NP}$ Leopold] [$_{VP}$ [$_V$ met] [$_{NP}$ e]$_1$]]

There is one piece of evidence for the assumption that the relation between the relative pronoun and the empty element, also called its **trace**, is comparable to the antecedent-pronoun relation discussed in the first section. There is another form of restrictive relative clause in English, often used in mathematics, in which the trace is actually an overt pronoun:

(29)Imagine a number such$_1$ that it$_1$ can be divided by seven.

The structure that is standardly assumed for relative clauses can be best illustrated with those more explicit forms:

(30)[$_{CP}$ [$_{NP}$ such]$_1$ [$_{C'}$ [$_{C0}$ that] [$_S$ it$_1$ can be devided by seven]]]

We assign the category CP to relative clauses. Following X-bar structure, they consist of one pronominal element (here, *such*) and a complementizer, category C^0 (here, *that*). Complementizers have a sentence S (or IP, following standard X-bar conventions) as a complement. In more standard types of relative clauses either the pronoun position or the complementizer position or both may be empty:

(31)a. woman [$_{CP}$ [$_{NP}$ e]$_1$ [$_{C'}$ [$_{C0}$ that] [$_S$ Stephen met e$_1$]]]
b. woman [$_{CP}$ [$_{NP}$ who]$_1$ [$_{C'}$ [$_{C0}$ e] [$_S$ Stephen met e$_1$]]]
c. woman [$_{CP}$ [$_{NP}$ e]$_1$ [$_{C'}$ [$_{C0}$ e] [$_S$ Stephen met e$_1$]]]

But for our current purposes we will work with a considerably simplified structure:

(32)woman [$_{CP}$ [$_C$ who]$_1$ [$_S$ Stephen met e$_1$]]

This is generated by the following syntactic rules:

(33)a. N → N CP
b. CP → C S

Of course we also must allow that NP's are interpreted as empty elements:

(34)NP → e

The next question is how these rules are interpreted so that we get a proper interpretation for relative clauses.

8.2.3 Interpretation of Relative Clauses

Let us now turn to the interpretation of relative clauses. I will concentrate first on appositive relative clauses of the structure (22).

The interpretation of the relative clause must ensure that the meaning of the empty element in the relative clause is used to form a complex predicate.

(35) ⟦[$_{CP}$ [$_C$ who]$_1$ [$_S$ e$_1$ snores]]⟧ ≈ λx[x snores]

How can this be accomplished? Notice that we have a coindexation between the head of the relative clause, *that,* and the empty element. This suggests that we can make use of variable assignments, which we have introduced to capture the relation between an antecedent and a pronoun.

Let's try. One plausible proposal is the following:

(36)a. ⟦[$_{CP}$ [$_C$ *who*]$_1$ [$_S$ e$_1$ *snores*]]⟧
b. = λg 1 DOM(g) λx D_e [g(1) snores, x = g(1)]

This is a function that maps variable assignments to functions from entities to truth values. In particular, it maps the assignment g to that function that maps an entity x to 1 if and only if Stephen met g(1), and x is g(1).

The interpretation of the relative marker *that* with index i that will give us this result is the following (I use T as a variable of type D_{at}, that is, functions from assignments to truth values).

(37) $[\![who_i]\!] = \lambda T \quad D_{at}\lambda g\ i \quad DOM(g) \quad \lambda x \quad D_e\ [T(g), x = g(i)]$

We can now construct the meaning of a relative clause as follows:

(38)a. $[\![[_{CP}\ [_{C}who]_1\ [_{CP}\ [_{NP}\ e]_1\ [_{VP}\ snores]]]]\!]$
b. $= [\![[_{C}who]_1]\!]([\![[_{CP}\ [_{NP}\ e]_1\ [_{VP}\ snores]]]\!]$
c. $= [\![[_{C}who]_1]\!]([\![[_{VP}\ snores]]\!]([\![[_{NP}\ e]_1]\!])$
d. $= [\![[_{C}who]_1]\!](\lambda g \quad G\lambda x \quad D_{et}[x\ \text{snores}](\lambda g\ 1 \quad DOM(g)\ [g(1)])$
e. $= [\![[_{C}who]_1]\!](\lambda g\ 1 \quad DOM(g)\ [g(1)\ \text{snores}])$
f. $= \lambda g\ 1 \quad DOM(g) \quad \lambda x \quad D_e\ [g(1)\ \text{snores}, x = g(1)]$

What we aimed for was the meaning $\lambda x[x\ \text{snores}]$. This is the result, once we factor in the variable assignment. For the condition "x = g(1)" means that we can replace g(1) by x, and so we end up with:

g. $= \lambda g\ 1 \quad DOM(g) \quad \lambda x \quad D_e\ [x\ \text{snores}]$

8.2.4 Variants of Variable Assignments

There is one weakness in the treatment of relative clauses in the preceding section. We have assumed so far that we can choose indices relatively freely, except that the relative clause marker *that* and an empty element must be coindexed. This would allow us to create structures like the following:

(39) *the man who*$_1$ e_1 *watches a woman who*$_1$ e_1 *snores*

We have here a relative clause within a relative clause. No problem with that, but in (39) we have reused an index within the domain of another index. Our rules don't exclude that, but they predict a rather weird meaning in this case. We haven't discussed quantified NPs like *a woman* in object position yet, and so we cannot discuss this example fully here. But we would get, essentially, the following derivation:

(40)a. $[\![$*woman who*$_1$ e_1 *snores*$]\!]$
$= \lambda g\ 1 \quad DOM(g) \quad \lambda x\ [x\ \text{is a woman}, [g(1)\ \text{snores}, x = g(1)]$
b. $[\![$*watches a woman who*$_1$ e_1 *snores*$]\!]$
$= \lambda g\ 1 \quad DOM(g) \quad \lambda y[\{x|y\ \text{watches}\ x\} \cap \{x|x\ \text{is a woman}, g(1)\ \text{snores}, x = g(1)\} \neq \quad]$
c. $[\![who_1\ e_1$ *watches a woman who*$_1$ e_1 *snores*$]\!]$
$= \lambda g\ 1 \quad DOM(g) \quad \lambda y[y = g(1),$
$\{x|y\ \text{watches}\ x\} \cap \{x|x\ \text{is a woman}, g(1)\ \text{snores}, x = g(1)\} \neq \quad]$

In step (b) we have created a predicate that applies to all entities y that watch a woman that snores. More specifically, we have a function that maps assignments g to functions from entities y to truth values, in particular, to the truth value 1 if there is an entity x such that y watches x and x is a woman that snores, and x is g(1).

In step (c) we have formed a relative clause out of that. According to our rules, we get a function that maps assignments g to functions from entities y to truth values. In particular, y is mapped to 1 if y = g(1) and if there is an entity y such that y watches x, x is a woman that snores, and x = g(1). The problem here is of course that both y, the watcher, and x, the watched, are said to be the

value of g applied to 1. This means that they must be the same. But this is not expressed at all in the relative clause *who watches a woman who snores.*

There are several possible remedies to that situation. One is to exclude it that a relative pronoun re-uses an index that was used by another relative pronoun. This would require some book-keeping of used indices which is possible, but perhaps a bit difficult to do.

The standard solution of dealing with this type of problem is to work with the notion of a **variant** of a variable assignment. A variant is an assignment that differs from another assignment in a particular way. We use the following notation:

(41)If g is an assignment, i is an index in its domain, and x is an entity, then
g[i/x] is that assignment that is just like g, except that it maps i to x.

For example,

(42)Let g = [1 → LB, 2 → MB, 3 → SD, 4 → LB], then
a. g[2/LB] = [1 → LB, 2 → LB, 3 → SD, 4 → LB]
b. g[2/MB] = g
c. g[2/LB][3/MB] = [1 → LB, 2 → LB, 3 → MB, 4 → LB]
d. g[2/LB][2/MB] = g

We now intepret the formation of relative clauses along the following lines:

(43) $[\![\textit{who}_i]\!] = \lambda T \quad D_{at} \lambda g\ i \quad DOM(g) \quad \lambda x \quad D_e\ [T(g[i/x])]$

(44)a. $[\![[_{CP}\ [_C \textit{who}]_1\ [_{CP}\ [_{NP}\ e]_1\ [_{VP}\ \textit{snores}]]]]\!]$
b. $= [\![\textit{who}]\!](\lambda g\ 1 \quad DOM(g)\ [g(1)\ \text{snores}])$
c. $= \lambda g\ 1 \quad DOM(g) \quad \lambda x \quad D_e\ [\lambda g \quad \{g \mid 1 \quad DOM(g)\}[g(1)\ \text{snores}](g[1/x])]$
d. $= \lambda g\ 1 \quad DOM(g) \quad \lambda x \quad D_e\ [g[1/x](1)\ \text{snores}]$
e. $= \lambda g\ 1 \quad DOM(g) \quad \lambda x \quad D_e\ [x\ \text{snores}]$

As for the last step, notice that g[1/x](1) is x, hence we can eliminate reference to indices altogher. They just served a useful purpose in constructing a particular meaning.

The problem we have observed with (39) does not occur anymore. We would now have the following derivation:

(45)a. $[\![\textit{woman who}_1\ e_1\ \textit{snores}]\!]$
$= \lambda g\ 1 \quad DOM(g) \quad \lambda x\ [x \text{ is a woman, } x \text{ snores}]$
b. $[\![\textit{watches a woman who}_1\ e_1\ \textit{snores}]\!]$
$= \lambda g\ 1 \quad DOM(g) \quad \lambda y[\{x \mid y \text{ watches } x\} \cap \{x \mid x \text{ is a woman, } x \text{ snores}\} \neq \quad]$
c. $[\![\textit{who}_1\ e_1\ \textit{watches a woman who}_1\ e_1\ \textit{snores}]\!]$
$= \lambda g\ 1 \quad DOM(g) \quad \lambda y[$
$\lambda g\ 1 \quad DOM(g)\ [\{x \mid g(1) \text{ watches } x\} \cap \{x \mid x \text{ is a woman, } x \text{ snores}\} \neq \quad](g[1/y])]$
$= \lambda g\ 1 \quad DOM(g) \quad \lambda y[\{x \mid g[1/y] \text{ watches } x\} \cap \{x \mid x \text{ is a woman, } x \text{ snores}\} \neq \quad]$
$= \lambda g\ 1 \quad DOM(g) \quad \lambda y[\{x \mid y \text{ watches } x\} \cap \{x \mid x \text{ is a woman, } x \text{ snores}\} \neq \quad]$

Notice that y, the watcher, and x, the watched, are not forced to refer to the same entity here.

8.3 Quantified Antecedents and Variable Binding

8.3.1 Pronouns and Quantified Antecedents

After our excursion into the world of relative clauses let us come back to the antecedent-pronoun relationship. In the examples we have considered so far a pronoun was interpreted as standing proxy for its antecedent. In essence, we interpreted (46) as indicated, using variable assignments to transfer to information that the index 1 had to be interpreted as LB.

(46)a. Leopold$_1$ scratched himself$_1$.
means: 'Leopold scratched Leopold'

This also works fine if the antecedent is a definite description, like *the man with the hat*. But in cases in which the antecedent is a quantified NP, the pronoun cannot be taken to stand proxy for its antecedent. First, take indefinite NPs:

(47)a. A man$_1$ scratched himself$_1$.
does not mean: A man scratched a man.
b. A man$_1$ loves his$_1$ wife.
does not mean: A man loves a man's wife.
c. A man$_1$ thinks that Molly loves him$_1$.
does not mean: A man thinks that Molly loves a man.

Take NPs with the universal quantifier *every*:

(48)a. Every man$_1$ scratched himself$_1$.
does not mean: Every man scratched every man.
b. Every man$_1$ loves his$_1$ wife.
does not mean: Every man loves every man's wife.
c. Every man$_1$ thinks that Molly loves him$_1$.
does not mean: Every man thinks that Molly loves every man.

Sometimes the paraphrases don't even make sense:

(49)No man$_1$ loves his$_1$ wife.
?? No man loves no man's wife.

Even if the antecedent is a name we sometimes cannot simply replace the pronoun by the name and retain the same truth conditions. Consider the following:

(50)a. Only Leopold loves his wife.
b. Only Leopold loves Leopold's wife.

Sentence (50.a) would be false if there is another married man that loves his (own) wife. Sentence (b) need not be false in this situation. It would be false if there is another person that loves *Leopolds'* wife. In this situation, in turn, sentence (a) need not be false.

8.3.2 Quantificational Prefixes

Even if pronouns with quantified antecedents are not really shorthands for their antecedents, the notion of a variable developed above can help us to understand the semantic mechanism by which these sentences are interpreted. Notice that these sentences allow for paraphrases of the following type. First, in the case of indefinite antecedents (47) we can give the following paraphrases:

(51)a. There is a man x such that x scratched x.
b. There is a man x such that x loves x's wife.
c. There is a man x such that x thinks that Molly loves x.

What we have done here is the following: We have replaced the antecedent and all the pronouns with the same index by the same variable, and we have used the antecedent to form a **quantificational prefix** that contains this variable that then was put in front of the sentence:

(52)a. Original sentence: [*A man*]$_1$ *scratched himself*$_1$.
b. Replacing by a variable: *x scratched x*.
c. Quantificational prefix: *There is a man x such that x scratched x.*

We say that the quantificational prefix **binds** the variable in question, and that the sentence to which it is attached is the **scope** of the quantificational prefix.

In the case of (48) we proceed in exactly the same way, except that the predicational prefix is different because the determiner in the antecedent is different:

(53)a. For every man x it holds: x scratched x.
b. For every man x it holds: x loves x's wife.
c. For every man x it holds: x thinks that Molly loves x.

In the case of an antecedent headed by *no*, as in (49), the prefix has a different form, "for no...":

(54)For no man x does it hold: x loves x's wife.

The way how paraphrases are derived are a bit less straightforward for (50). Here they are:

(55)a. For every x such that x loves x's wife: x is Leopold.
b. For every x such that x loves Leopold's wife: x is Leopold.

8.3.3 Quantificational Prefixes and Generalized Quantifiers

In chapter (6) we have developed a theory of quantificational NPs in English in which they are analyzed as expressions with meanings of type (et)t that take one-place predicates, type et, as their arguments. We had interpretations of the following type:

(56)a. $[\![a\ man]\!] = \lambda P[[\![man]\!] \cap P \neq \emptyset]$
b. $[\![every\ man]\!] = \lambda P[[\![man]\!] \subseteq P]$
c. $[\![no\ man]\!] = \lambda P[[\![man]\!] \cap P = \emptyset]$

How can we understand the quantificational prefixes that we have used in the paraphrases in section (8.3.2), given these meanings of quantifiers? It helps to reformulate the paraphrases in the following way:

(57)a. A man has the following property: to be an x such that x scratches x.
b. Every man has the following property: to be an x such that x scratches x.
c. No man has the following property: to be an x such that x scratches x.

The first part, e.g. *every man has the following property:*, comes close to the meaning of the quantifier *every man*, which describes a certain property P (namely, by saying that P is a subset of $[\![man]\!]$). We also know, roughly, how to deal with the second part. It is a description of a property,

for which we can make use of the technique of lambda abstraction. The analysis we are heading towards can be sketched as follows:

(58)a. ⟦*every man*$_1$ *scratched himself*$_1$⟧
b. = λP[λx[x is a man] P](λy[y scratched y])
c. = λx[x is a man] λy[y scratched y]

But there are still two important tasks to be accomplished. First, how do we derive the interpretation indicated in (58.b) from (a)? Second, how can we derive the interpretation rendered as λy[y scratched y] in a systematic way? We will address these issues in turn.

8.3.4 Quantifier Raising

We have to interpret the following syntactic structure:

(59)[$_S$ [$_{NP}$ [$_{Det}$ *every*] [$_N$ *man*]]$_1$ [$_{VP}$ [$_V$ *scratched*] [$_{NP}$ *himself*]$_1$]]

One way to achieve the intepretation indicated in (58.b) is to form the meaning of a relative clause, with the empty element in the position of the quantified NP *every man*, and then apply the meaning of the quantified NP to the relative clause meaning:

(60)a. ⟦*every man*⟧(⟦*who*$_1$ [$_S$ e$_1$ *scratched himself*$_1$]⟧)
b. = λP[⟦*man*⟧ P](λx[x scratched x])

This does not mean that we indeed form a relative clause; notice that a phrase like *every man who scratched himself* would be interpreted quite differently, with the relative clause modifying the noun *man*. It is rather suggested that when we interpret (59) we perform a task that is equivalent to relative clause formation.

The essential step in relative clause formation is that an argument of a sentence represented by a relative pronoun is moved from its original position and attached to the whole sentence, leaving an empty element:

(61)a. [$_S$ [$_{NP}$ who]$_1$ [$_{VP}$ [$_V$ scratched] [$_{NP}$ himself]$_1$]]
b. [who]$_1$ [$_S$ e$_1$ [$_{VP}$ [$_V$ scratched] [$_{NP}$ himself]$_1$]]

We can assume that the same happens with quantified noun phrases. That is, they attach to the whole sentence, leaving a trace in their original position.

(62)a. [$_S$ [$_{NP}$ every man]$_1$ [$_{VP}$ [$_V$ scratched] [$_{NP}$ himself]$_1$]]
b. [every man]$_1$ [$_S$ e$_1$ [$_{VP}$ [$_V$ scratched] [$_{NP}$ himself]$_1$]]

This movement of the quantified noun phrase is not visible in the syntactic structure itself, and thus different from the overt movement we find with relative pronouns. Compare the following two cases which involve the object argument:

(63)a. who$_1$ Leopold met e$_1$
b. Leopold met every woman$_1$

But we can assume that between visible syntax and semantic interpretation there are "invisible" syntactic operations. In the framework called **Government and Binding** (Chomsky 1981) it was assumed that we distinguish between a level of **S-Structure** ("surface structure") and a level of **Logical Form** (LF). LF is seen as the input to semantic interpretation. The specific operation of

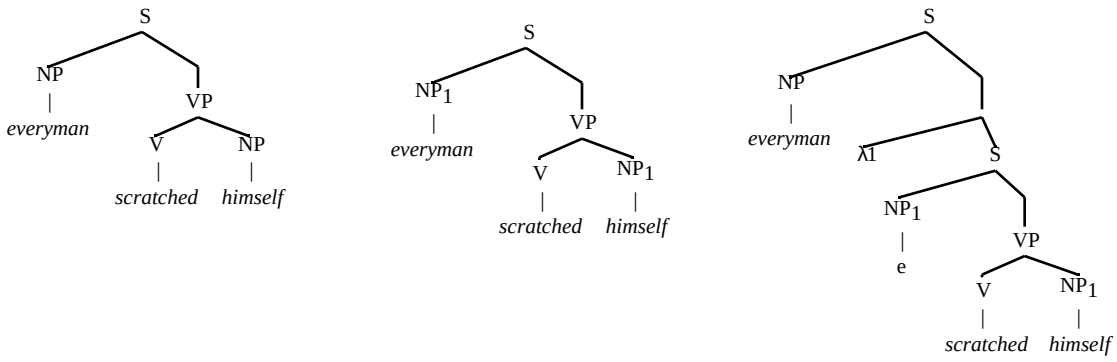
moving a quantifier to attach to a sentence while leaving a coindexed trace is called **quantifier raising**, after May (1977).

We will assume here the following format for quantifier raising:

(64)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.

We assume here that all NPs come with an index. Pronouns bear the same index as their antecedent, and NPs that do not stand in an anphoric relationship have different indices. To indicate how a derived syntactic structure is interpreted, I have included a lambda plus an index in the logical form. This is not quite how logical forms are usually given; in the usual format, the noun phrase typically keeps its index instead. The way how the LF of *every man scratched himself* is derived is then as follows:

(65)



S-Structure **Indexed S-Structure** **Quantifier Raising**

We have the following interpretation rule, which is essentially the same as for relative clauses:

(66) $[\![\lambda i [_S \alpha]]\!] = \lambda g\ i\ \ DOM(g)\ \ \lambda x\ x\ \ D_e, g[i/x]\ \ DOM([\![\alpha]\!])\ \ [[\![\alpha]\!](g[i/x])]$

And we have derivations like the following:

(67)a. S-Structure: $[_S [_{NP} [_{Det} \textit{every}] [_N \textit{man}]]_1 [_{VP} [_V \textit{scratched}] [_{NP} \textit{himself}]_1]]$

b. LF: $[_{NP} [_{Det} \textit{every}] [_N \textit{man}]] \lambda 1 [_S [_{NP} e]_1 [_{VP} [_V \textit{scratched}] [_{NP} \textit{himself}]_1]]$

c. $[\![[_{NP} [_{Det} \textit{every}] [_N \textit{man}]] \lambda 1 [_S [_{NP} e]_1 [_{VP} [_V \textit{scratched}] [_{NP} \textit{himself}]_1]]]\!]$

d. $= \lambda g\ \ DOM([\![[_{NP} [_{Det} \textit{every}] [_N \textit{man}]]]\!]) \cap DOM([\![\lambda 1 [_S [_{NP} e]_1 [_{VP} [_V \textit{scratched}] [_{NP} \textit{himself}]_1]]]\!])$
$[\![[_{NP} [_{Det} \textit{every}] [_N \textit{man}]]]\!](g)([\![\lambda 1 [_S [_{NP} e]_1 [_{VP} [_V \textit{scratched}] [_{NP} \textit{himself}]_1]]]\!](g))$

e. $[\![[_{NP} [_{Det} \textit{every}] [_N \textit{man}]]]\!]$
$= \lambda g\ \ G[[\![\textit{every}]\!](g)([\![\textit{man}]\!](g)]$
$= \lambda g\ \ G \lambda P[\lambda x[x \text{ is a man}]\ \ P]$

f. $[\![\lambda 1[_S\ [_{NP}\ e]_1\ [_{VP}\ [_V\ \textit{scratched}]\ [_{NP}\ \textit{himself}]_1]]\!]$

g. $= \lambda g\ 1 \in DOM(g)\ \lambda x\ x \in D_e, g[1/x] \in DOM([\![[_S\ [_{NP}\ e]_1\ [_{VP}\ [_V\ \textit{scratched}]\ [_{NP}\ \textit{himself}]_1]]\!])$
$[\![[_S\ [_{NP}\ e]_1\ [_{VP}\ [_V\ \textit{scratched}]\ [_{NP}\ \textit{himself}]_1]]\!](g[1/x])]$

h. $[\![[_S\ [_{NP}\ e]_1\ [_{VP}\ [_V\ \textit{scratched}]\ [_{NP}\ \textit{himself}]_1]]\!]$

i $= \lambda g \in DOM([\![[_{NP}e]_1]\!]) \cap DOM([\![[_{VP}[_V\textit{scratched}]\ [_{NP}\textit{himself}]_1]]\!])$
$[\![[_{VP}[_V\textit{scratched}]\ [_{NP}\textit{himself}]_1]]\!](g)([\![[_{NP}e]_1]\!](g))]$

k $= \lambda g \in DOM([\![[_{NP}e]_1]\!]) \cap DOM([\![[_V\textit{scratched}]]\!]) \cap DOM([\![[_{NP}\textit{himself}]_1]\!])$
$[[\![\textit{scratched}]\!](g)([\![[_{NP}\textit{himself}]_1]\!])(g))([\![[_{NP}e]_1]\!](g))]$

l. $= \lambda g\ 1 \in DOM(g), g \in G, g(1)$ is male
$[\lambda x \lambda y[y$ scratched $x](g(1))(g(1))]$

m. $= \lambda g\ g(1)$ is male $[g(1)$ scratched $g(1)]$

n. $DOM(\lambda g\ g(1)$ is male $[g(1)$ scratched $g(1)]) = \{g \in G \mid g(1)$ is male$\}$

o. line g $= \lambda g\ 1 \in DOM(g)\ \lambda x\ g[1/x] \in \{g \in G \mid g(1)$ is male$\}$
$[\lambda g\ g(1)$ is male $[g(1)$ scratched $g(1)](g[1/x])]$

p. $= \lambda g\ 1 \in DOM(g)\ \lambda x\ x$ is male $[\lambda g\ g(1)$ is male $[g(1)$ scratched $g(1)](g[1/x])]$

q. $= \lambda g\ 1 \in DOM(g)\ \lambda x\ x$ is male $[g[1/x]$ is male $\mid g[1/x](1)$ scratched $g[1/x](1)]$

r. $= \lambda g\ 1 \in DOM(g)\ \lambda x\ x$ is male $[x$ scratched $x]$

s. line d $= \lambda g\ 1 \in DOM(g)\ [\lambda x[x$ is a man$]\ \lambda x\ x$ is male $[x$ scratched $x]$

We get the right result here: A function from assignments that is mapped to 1 if the set of men is a subset of the male self-scratchers; and else to 0. Notice how, according to rule (66), the a condition on the value of g(1) is passed over to a condition on the variable x, by way of the assignemnt variant g[1/x] (cf. steps (n) to (r) in the above derivation).

Other cases with quantified NPs as antecedents can be treated in a completely parallel fashion.

8.3.5 Pronoun Binding by Names

We started the discussion in this chapter by observing that, if the antecedent of a pronoun is a name, then the pronoun just can be spelled out as this name. This allows for paraphrases like the following:

(68)a. Leopold$_1$ loves his$_1$ wife.
'Leopold loves Leopold's wife.'
b. Leopold thinks that Molly loves him.
'Leopold thinks that Molly loves Leopold.'

Now, we have seen in chapter (6) that we can treat names as quantifiers. For example, we can type-raise LB to $\lambda P \in D_{et}[P(LB)]$. But then we can assume that names bind their pronouns as variables, just like quantifiers.

Let us see how we can derive example (68.a). For that we first have to determine the interpretation of the possessive pronoun *his*. We can intepret *his wife* as *the wife of him*. For our purposes it is sufficient to assume the following interpretation:

(69) $[\![his_1]\!] = \lambda g\ g(1)$ is male $\lambda R\ D_{eet}\ \#\{y \mid R(g(1))(y) = 1\}\ [\iota y[R(g(1))(y)]$

That is, his_1 combines with a relational noun meaning R. It presupposes that g(1) is male, and that there is exactly one y that stands in R-relation to g(1). And it gives that y as a value. The phrase his_1 *wife* then refers to the wife of g(1):

(70) $[\![[_{NP}[_{Det} his_1]\ [_N wife]]]\!]$
= $\lambda g\ g(1)$ is male, $\#\{y \mid y$ is wife of $g(1)\} = 1\}\ [\iota y[y$ is wife of $g(1)]]$

Example (68.a) now can be either interpreted with the LF in (71), where the name stays in situ, or as in (72), where the name undergoes quantifier raising:

(71)$[_S\ [_{NP}\ Leopold]_1\ [_{VP}\ [_V\ loves]\ [_{NP}\ [_{Det}\ his]_1\ [_N\ wife]]]]$

(72)$[_S\ [_{NP}\ Leopold]\ \lambda 1[e_1\ [_{VP}\ [_V\ loves]\ [_{NP}\ [_{Det}\ his]_1\ [_N\ wife]]]]]$

In the first case we get the following interpretation:

(71)a. $[\![[_S\ [_{NP}\ Leopold]_1\ [_{VP}\ [_V\ loves]\ [_{NP}\ [_{Det}\ his]_1\ [_N\ wife]]]]]\!]$
b. $= \lambda g[[\![[_{VP}\ [_V\ loves]\ [_{NP}\ [_{Det}\ his]_1\ [_N\ wife]]]]\!](g)([\![Leopold_1]\!](g))$
c. $= \lambda g[[\![loves]\!](g)([\![[_{NP}\ [_{Det}\ his]_1\ [_N\ wife]]]\!](g))([\![Leopold_1]\!](g))$
d. $= \lambda g[\lambda g \lambda y \lambda x[x$ loves $y](g)$
$(\lambda g\ g(1)$ is male, $\#\{y \mid y$ is wife of $g(1)\} = 1\}\ [\iota y\ [y$ is wife of $g(1)]](g))$
$(\lambda g\ g(1) = LB\ [g(1)](g))]$
e. $= \lambda g\ g(1)$ is male, $g(1) = LB$, $\#\{y \mid y$ is wife of $g(1)\} = 1$
$[g(1)$ loves $\iota y[y$ is wife of $g(1)]]$

That is a function from assignments that map g(1) to a male entity, and in particular to Leopold, and for which it holds that g(1) has exactly one wife, to the value 1 (true) if g(1) loves g(1)'s wife.

In the second case we have the following interpretation:

(72)a. $[\![[_S\ [_{NP}\ Leopold]\ \lambda 1[e_1\ [_{VP}\ [_V\ loves]\ [_{NP}\ [_{Det}\ his]_1\ [_N\ wife]]]]]\!]$
b. $= \lambda g[[\![Leopold]\!](g)([\![\lambda 1[e_1\ [_{VP}\ [_V\ loves]\ [_{NP}\ [_{Det}\ his]_1\ [_N\ wife]]]]]\!](g))$
c. (several steps)
d. $= \lambda g[\lambda P[P(LB)](\lambda x\ x$ is male, $\#\{y \mid y$ is wife of $x\} = 1\ [x$ loves $\iota y[y$ is wife of $x]])]$
e. $= \lambda g\ LB$ is male, $\#\{y \mid y$ is wife of $LB\} = 1\ [LB$ loves $\iota y[y$ is wife of $x]]$

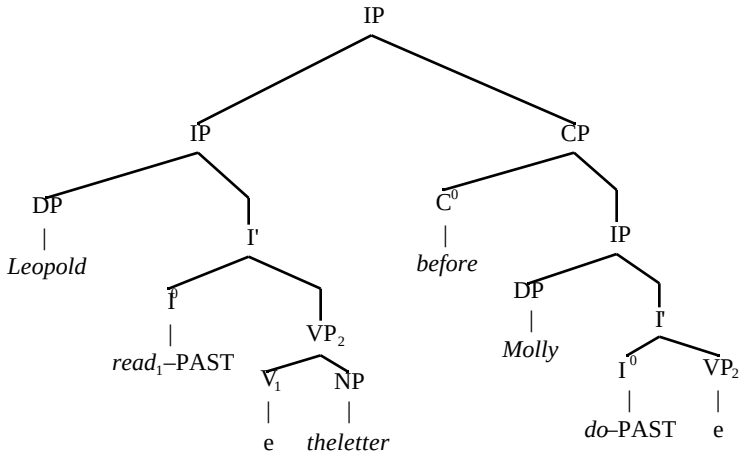
We get essentially the same information: It is presupposed that Leopold is male and has exactly one wife, and it is expressed that he love his wife.

Is there any evidence that could decide in which way we interpret pronouns with names as their antecedent? Actually, there is reason to believe that we can go either way. The crucial phenomenon in this respect is a type of anaphora we haven't considered so far, namely, **VP ellipsis**.

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

The *did* in these examples has to be spelled out as *read the letter*. Hence VP ellipsis is a kind of anaphora, just like pronouns. Using the notions of X-bar theory developed in chapter (3), we can analyze (73.b) as follows:

(74)



The first trace, e_1, has to be interpreted as *read*; the verb has moved to the Inflection node to be combined with the past tense marker, which gives us the verb form *read*. The empty VP in the second clause has to be interpreted as its antecedent in the first class, which is, after we replace e_1 by its antencedent, the VP *read the letter*. The verb *do* can be seen as a "dummy" verb that has to be there because tense has to be expressed. We predict the correct reading of (73.b), namely, 'Leopold read the letter before Molly read the letter'.

Now let us consider the following description of a scene of *Romeo and Juliet*:

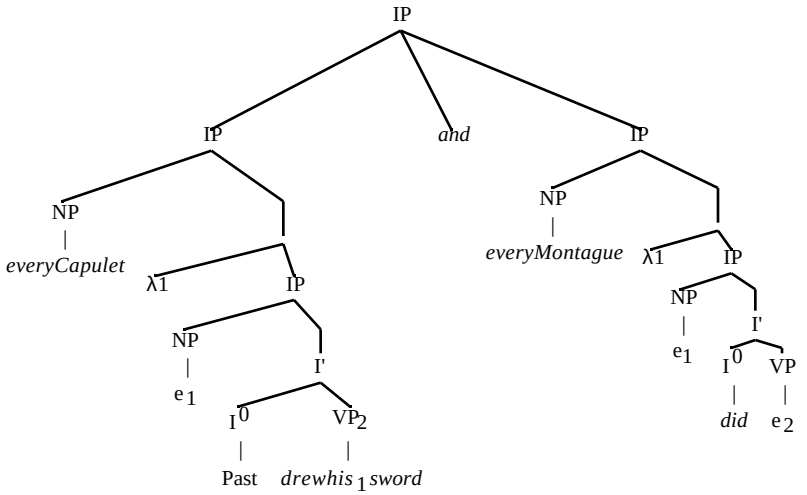
(75)Every Capulet drew his sword, and every Montague did, too.

We typically understand this as follows:

'For every Capulet x: x drew x's sword, and for every Montague x: x drew x's sword.

The VP ellipsis has to be spelled out as *drew his sword*, but *his* has to be bound by *every Montague* here, not by *every Capulet*. Our theory predicts that if we allow for the following interpretation:

(76)



The empty VP has to be interpreted like his antecedent, *drew his*$_1$ *sword*. (We will not try to implement this here; we would have to use a notion of variable assignment that is not restricted to entities of type e, but also allows for variables for meanings of type et.) Then it is quite clear that we will get the right interpretation. Whithout going into details here, it will be something like:

(77) λP[Capulet P](λx[x drew x's sword]) and λP[Montague P](λx[x drew x's sword])

Now let us consider a case that involves two names:

(78)Tybalt drew his sword, and Mercutio did, too.

The preferred reading of the second clause is clearly that Mercutio drew Mercutio's sword. The only way to generate this reading, under the assumption that VP ellipsis is a shorthand for its antecedent, is to assume that names undergo quantifier raising. Then we get a similar structure as in (76).

In other examples the other reading appears to be more prominent:

(79)Mercutio pressed against his wound, and the nurse did, too.

Here we get the intepretation that the nurse pressed against Mercutio's wound. This has been called the **strict** interpretation, whereas the interpretation of (78) is called the **sloppy** interpretation. The strict interpretation can be generated, of course, when the name is interpreted as being of type e and does not undergo quantifier raising.

8.4 Other Types of Anaphora

8.4.1 Deictic Pronouns

There are a number of other uses of pronouns that should be mentioned here. One is the **deictic** use of pronouns. The pronoun lacks any linguistic antecedent, but it is clear from the context of utterance which entity the pronoun refers to. We find this use, for example, if a speaker says the following, with the appropriate pointing gestures:

(80)He beat him first.

There are certain kinds of pronouns that are always deictic, namely the pronouns referring to the speaker and the addresse of a sentence, *I* and *we*, and *you.*

We can incorporate this use of pronouns as follows: The set of variable assignments G contains information about the **context** in which the sentence is uttered. For example, if a sentence that contains the indexed pronoun he_{17} is uttered in a context, then every assignment g that can be used to interpret the sentence contains the information that g(17) refers to the person meant by $he_{17.}$

This notion of context obviously has to be developed further, which will be done in the continuation of this course.

8.4.2 Cross-Sentential Anaphora

The antecedent of a pronoun can be located in a preceding sentence in a text:

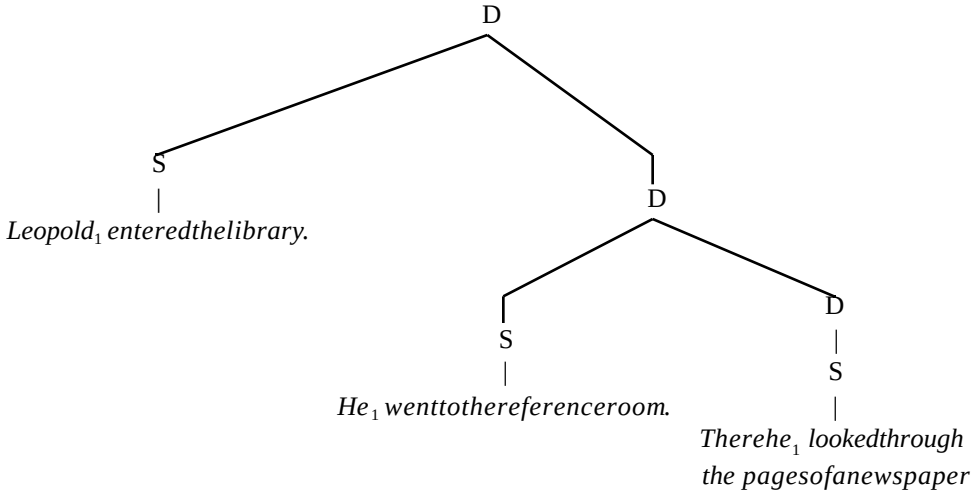
(81)Leopold$_1$ entered the library. He$_1$ went to the reference room. There he$_1$ looked through the pages of a newspaper.

We can deal with such cases in a rather straightforward way. First, we have to assume a rule that allows us to build texts out of sentences. To do this job properly would require considerable attention to the structuring of texts — in units like chapters or paragraphs, which are not only important for written text, but also for spoken discourse. For our current purposes we can regard texts as sequences of texts. The following two phrase-structure rules will do that; I use the category symbol "D" for "discourse".

(82)a. D → S D
b. D → S

This will give us the following structure for (81):

(83)



How should the sentences of a text be interpreted? A natural choice is by **conjunction**; a discourse is true iff all its sentences are true. So we should assume the following interpretation rule:

(84) $[\![[_D\ \alpha\ \beta]]\!] = \lambda g\ g \in DOM([\![\alpha]\!]), g \in DOM([\![\beta]\!])\ [[\![\alpha]\!](g), [\![\beta]\!](g)]$

Let us consider a simple example:

(85)a. $[\![[_D\ [_S\ [_{NP}\ \textit{Leopold}]_1\ [_{VP}\ \textit{came in}]]\ [_D\ [_S\ [_{NP}\ \textit{He}]_1\ [_{VP}\ \textit{sat down}]]]]]\!]$

b. $= \lambda g\ g \in DOM([\![[_S\ [_{NP}\ \textit{Leopold}]_1\ [_{VP}\ \textit{came in}]]]\!]), g \in DOM([\![[_S\ [_{NP}\ \textit{He}]\ [_{VP}\ \textit{sat down}]]]\!])$
$[[\![[_S\ [_{NP}\ \textit{Leopold}]_1\ [_{VP}\ \textit{came in}]]]\!](g), [\![[_S\ [_{NP}\ \textit{He}]\ [_{VP}\ \textit{sat down}]]]\!](g)]$

c. $[\![[_S\ [_{NP}\ \textit{Leopold}]_1\ [_{VP}\ \textit{came in}]]]\!] = \lambda g\ g(1) = LB$ [g(1) came in]

d. $[\![[_S\ [_{NP}\ \textit{He}]\ [_{VP}\ \textit{sat down}]]]\!] = \lambda g\ 1 \in DOM(g)$ [g(1) sat down]

e. (b) $= \lambda g\ g(1) = LB, 1 \in DOM(g)$ [g(1) came in], [g(1) sat down]]

We get a function that maps variable assignments g, under the condition that g(1) is Leopold, to the truth value 1 if both g(1) came in and g(1) sat down is true, and else to 0. This is the correct result.

Interestingly, our semantic framework also suggests that the following sentence cannot mean that every man came in and sat down.

(86)Every man came in. He sat down.

That is, *every man* cannot be antecedent of *he*. Even if we use the same indices, *he* is not bound by *every man.* The reason is that we have formulated the interpretation rule of quantifiers in such a way that *every man* can scope just over its own sentence. To be specific, *every man* will undergo quantifier raising, which leads to the following structure and interpretation:

(86)a. $[_D\ [_S\ [_{NP}\ \textit{every man}]\ \lambda 1 [_S\ e_1\ \textit{came in}]]\ [_D\ [_S\ \textit{He}_1\ \textit{sat down}]]]$

b. $[\![[_S\ [_{NP}\ \textit{every man}]\ \lambda 1 [_S\ e_1\ \textit{came in}]]]\!]$
$= \lambda g\ 1 \in DOM(g)$ λx[x is a man] λx[x xame in]]

c. $[\![[_S\ \textit{He}_1\ \textit{sat down}]]\!] = \lambda g\ 1 \in DOM(g)$ [g(1) sat down]

d. $[\![(a)]\!] = \lambda g\ 1 \in DOM(g)$ [λx[x is a man] λx[x xame in]], [g(1) sat down]]

Clearly, there is no connection between the men we quantify over and the meaning of $\textit{he}_1$, which is g(1). The pronoun can only be interpreted as a deictic pronoun, referring to some man given in the context.

However, this holds only for some NPs that we analyzed as quantifiers. For indefinite NPs we find anaphora across sentence boundaries:

(87)A man_1 came in. He_1 sat down.

This means that contrary to our analysis in chapter (6), indefinite NPs have to be interpreted quite differently. They have been the object of intensive study, in particular starting with the work of Hans Kamp (1981) and Irene Heim (1982).

One strategy in dealing with indefinite NPs has been to assume that they "extend" a variable assignment by introducing a new index. It is convenient here to talk about an **input** variable assignment and an **output** variable assignment. In (87), the first sentence expects an input variable assignment for which the index 1 is not defined yet, and yields a variable assignment for which the index 1 is defined, and refers to some man or other. This output of the first sentence is used as an input to the second sentence. The pronoun $\textit{he}_1$ expects that the index 1 is defined, and in the context of the first sentence this is indeed the case.

Another strategy is to see the pronoun as a concealed definite description. In our example, $\textit{he}_1$ is somehow spelled out as *the man who came in.* This strategy goes back to Robin Cooper (1978) and Garrett Evans (1977).

8.4.3 Donkey Anaphora

Cross-sentential anaphora is not the only domain in which indefinite NPs show a special behavior. They are also peculiar in the following cases, which are called “donkey sentences”, after examples that date back to medieval logicians:

(88)a. Every farmer$_1$ who owns a donkey beats it$_1$.
b. Every man$_1$ that came in lifted his$_1$ hat.

It is clear what these sentences should mean. For example, (a) can be paraphrased as: For every farmer x and every donkey y such that x owns y it holds that x beats y. However, our current analysis of indefinites as quantifiers that assert the existence of an entity do not give us that as a result. We haven’t treated quantifiers in object positions yet (we will do this in chapter 10), but essentially the only way how *a donkey* can get into a position from which it can bind the pronoun *it* is by LF-movement out of the relative clause. The subject quantifier moves as well, which results in one of the following two structures:

(89)a. [[a farmer] λ2[[every donkey] λ1[[e$_1$ [e$_1$ owns e$_2$]] beats e$_2$]]]
b. [[every donkey] λ1[[a farmer] λ2[[e$_1$ [e$_1$ owns e$_2$]] beats e$_2$]]]

The first interpretation is ‘there is a farmer x and every donkey y such that x owns y is beaten by x’, clearly not the intended reading. The second interpretation says ‘every donkey y is owned by a farmer x, and x beats y’, which is again not what our example means.

The possible solutions to the problem of donkey anaphora are quite similar to the solution of intra-sentential anaphora. That is, we either make use of input assignments and output assignments, or we interpret the pronoun as a definite description in disguise. The subject of the interpretation of pronouns will be taken up in the continuation of this course.