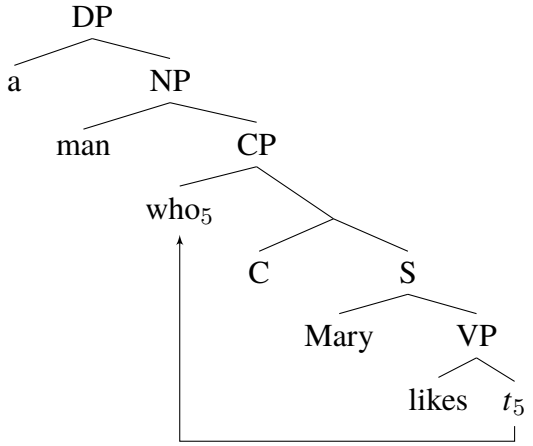
1 Relative Clauses in English

In this lecture we will analyze **relative clauses**, which function as nominal modifiers, just like adjectives. In English, you can find several types of relative clauses.

(1) a. a man who Mary likes (*wh*-relative)
b. a man that Mary likes (*that*-relative)
c. a man Mary likes (zero relative)

For the moment let us focus on the first type, *wh*-relatives. This type of relative clauses involve a *wh*-phrase like 'who', which is called a relative pronoun. As you probably learned in syntax courses, relative pronouns undergo movement and leave a trace in the base-generated position. This is illustrated by the tree diagram in (2). We assume that there is a complementizer C, although it is not phonologically realized.

(2)



The relative pronoun has an index 5 in (2), and its trace has the same index 5. As we will see, these indices play an important role in the compositional semantics.

What do relative clauses denote? As remarked above, relative clauses are modifiers of nouns, on a par with adjectives. This becomes clear if you compare the following pair.

(3) a. boy who is blond
b. blond boy

These noun phrases mean essentially the same thing, so we want them to have the same denotation, which is to say:

$$[\![\text{blond boy}]\!]^{a,M} = [\![\text{boy who is blond}]\!]^{a,M}$$

Recall from the previous lecture that we assume $[\![\text{blond}]\!]^{a,M}$ is a function of type $\langle e,t\rangle$ and modifies $[\![\text{boy}]\!]^{a,M}$ via Predicate Modification. This is shown in the top-down following computation.

$$\left[\!\!\left[\begin{array}{c} \diagup\diagdown \\ \text{blond}\quad\text{boy} \end{array}\right]\!\!\right]^{a,M}$$

$= [\lambda x \in D_e.\ [\![\text{blond}]\!]^{a,M}(x) = 1 \text{ and } [\![\text{boy}]\!]^{a,M}(x) = 1]$ (PM)

$= [\lambda x \in D_e.\ [\lambda y \in D_e.\ 1 \text{ iff } y \text{ is blond in } M](x) = 1 \text{ and } [\![\text{boy}]\!]^{a,M}(x) = 1]$ (Lexicon)

$= [\lambda x \in D_e.\ x \text{ is blond in } M \text{ and } [\![\text{boy}]\!]^{a,M}(x) = 1]$ (λ-conv.)

$= [\lambda x \in D_e.\ x \text{ is blond in } M \text{ and } [\lambda z \in D_e.\ 1 \text{ iff } z \text{ is a boy in } M](x) = 1]$ (Lexicon)

$= [\lambda x \in D_e.\ x \text{ is blond in } M \text{ and } x \text{ is a boy in } M]$ (λ-conv.)

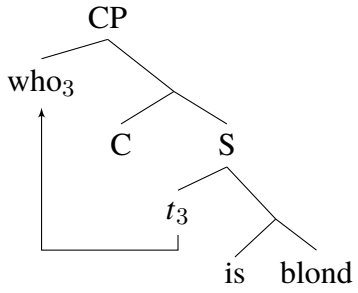
We want (3a) with a relative clause to have the same denotation, so we want to have:

$$[\![\text{boy who is blond}]\!]^{a,M} = [\lambda x \in D_e.\ x \text{ is blond in } M \text{ and } x \text{ is a boy in } M]$$

How can we achieve this?

Let us examine the structure of the relative clause:

(4)

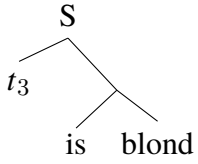


We know the denotations of 'is' and 'blond', but we don't know the denotations of the trace 't_3', the relative pronoun 'who$_3$' and the complementizer C. Let us discuss them in turn.

2 Traces as Variables

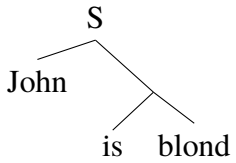
Let us zoom in on the S node in the diagram (4).

(5)



This looks a lot like what we are already familiar with, e.g. (6).

(6)



If the trace t_3 denotes an individual (i.e. is of type e), just like proper names, the computation of (5) will go through in exactly the same way as (6). That traces denote individuals is a plausible hypothesis, given that they generally appear in places where you can find proper names. Here is another example illustrating this point.

(7) a. Mary likes John.
b. John is a boy who$_8$ Mary likes t_8.

So if traces denote individuals, just like proper names, we don't need to worry about type-mismatches.

But if traces are of type e, which individuals should they denote? A moment's reflection reveals that they should not have a fixed referent, unlike proper names. Specifically, the trace in (8a) below might refer to John, but it shouldn't in (8b) and (8c), because these sentences are simply not about John.

(8) a. John is a boy who$_3$ Mary likes t_3.
b. Fred is a boy who$_3$ Mary likes t_3.
c. Oliver is a boy who$_3$ Mary likes t_3.

Rather, the denotations of traces should vary according to the syntactic environment they occur in. In (8a), t_3 denotes John, but (8b) it denotes Fred and in (8c) it denotes Oliver. To capture this, we analyze traces as **variables**.

Variables do not have denotations on their own. Rather, their denotations depend on an **assignment function** (sometimes simply called an **assignment**). An assignment function a is a function that maps indices to individuals. Since we assume here that indices are natural numbers, we can say an assignment function is a function from natural numbers to individuals in the model, which we can write as, $a\colon\ \mathbb{N} \rightarrow D_e$.[1]

The idea is to relativize the denotation of a trace to an assignment function. For example, we have something like (9).

(9) a. $[\![t_2]\!]^{a,M} = a(2) = \text{John}$
b. $[\![t_{101}]\!]^{a,M} = a(101) = \text{Mary}$

$[\![\alpha]\!]^{a,M}$ is the denotation of expression α in model M relative to assignment function a. We can have a different assignment function, which might return a different individual for the same index.

(10) a. $[\![t_2]\!]^{a,M} = a(2) = \text{John}$
b. $[\![t_2]\!]^{b,M} = b(2) = \text{Adam}$

So the assignment function determines the denotations of traces. Let us state this as a compositional rule.

(11) *Trace Rule*: For any model M and for any assignment function a and for any index $i \in \mathbb{N}$, $[\![t_i]\!]^{a,M} = a(i)$.

From now on, we will always consider denotations relative to some assignment function. The denotations of lexical items that are not traces are not affected by assignment functions. Specifically, the denotations of proper names, verbs, nouns, etc. are only dependent on the model. So whatever a, b and c might be, we have:

(12) a. $[\![\text{John}]\!]^{a,M} = [\![\text{John}]\!]^{b,M} = [\![\text{John}]\!]^{c,M}$
b. $[\![\text{smokes}]\!]^{a,M} = [\![\text{smokes}]\!]^{b,M} = [\![\text{smokes}]\!]^{c,M} = [\lambda x \in D_e.\ 1 \text{ iff } x \text{ smokes in } M]$

Similarly, the denotations of complex phrases that do not contain traces are insensitive to the assignment function (though they are generally sensitive to the model M).

(13) $[\![\text{John loves Mary}]\!]^{a,M} = [\![\text{John loves Mary}]\!]^{b,M} = 1$ iff John loves Mary in M

Only traces and phrases containing them are assignment-dependent. Concretely, let us assume

[1]Recall '$f\colon\ A \rightarrow B$' means 'f is a function whose domain is A and whose range is B.

the following two assignment functions, a and b. Recall that assignment functions are functions from indices (=natural numbers) to individuals.

(14) $$a = \begin{bmatrix} 1 & \rightarrow & \text{John} \\ 2 & \rightarrow & \text{John} \\ 3 & \rightarrow & \text{Mary} \\ 4 & \rightarrow & \text{Bill} \\ 5 & \rightarrow & \text{Daniel} \\ \dots & & \end{bmatrix} \qquad b = \begin{bmatrix} 1 & \rightarrow & \text{Mary} \\ 2 & \rightarrow & \text{John} \\ 3 & \rightarrow & \text{Bill} \\ 4 & \rightarrow & \text{Kate} \\ 5 & \rightarrow & \text{Hanna} \\ \dots & & \end{bmatrix}$$

For example, we have:

$$a(2) = \text{John} \quad b(2) = \text{John}$$
$$a(5) = \text{Daniel} \quad b(5) = \text{Hanna}$$

Then,

$$[\![t_2]\!]^{a,M} = [\![t_2]\!]^{b,M} = \text{John}$$
$$[\![t_2 \text{ loves Mary}]\!]^{a,M} = [\![t_2 \text{ loves Mary}]\!]^{b,M} = 1 \text{ iff John loves Mary in } M$$

but

$$[\![t_5]\!]^{a,M} \neq [\![t_5]\!]^{b,M}$$
$$[\![t_5 \text{ loves Mary}]\!]^{a,M} = 1 \text{ iff Daniel loves Mary in } M$$
$$[\![t_5 \text{ loves Mary}]\!]^{b,M} = 1 \text{ iff Hanna loves Mary in } M$$

Recall now that our goal here is to give the following denotation to the relative clause 'who_3 t_3 is blond' (the structure is omitted here to save space):

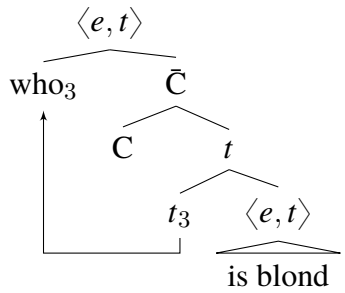
(15) $$[\![\text{who}_3\ t_3 \text{ is blond}]\!]^{a,M} = [\lambda x \in D_e.\ 1 \text{ iff } x \text{ is blond in } M]$$

Then, we want t_3 to be interpreted as a variable, x here, which is bound by λx in this representation, rather than a particular individual like John and Daniel. In order to achieve this, we need to say something about the other two ingredients of the relative clause, namely, the complementizer C and the relative pronoun.

3 Complementizer

If the whole relative clause is of type $\langle e,t\rangle$, as in (15), and the trace is of type e, we have the following semantic types:

(16)



In particular, the sister to C is of type t, but the entire relative clause is of type $\langle e,t\rangle$.

For the complementizer C, we simply assume that it is semantically vacuous, as it seems to add no meaning. Recall that semantically vacuous items denote identity functions. To give a denotation to C, all we need to know is its semantic type. Since its sister constituent is a sentence and is of type t, C should denote the identity function of type $\langle t,t\rangle$.

(17) For any model M and for any assignment function a,
$[\![\mathrm{C}]\!]^{a,M} = [\lambda v \in D_t.\ v]$

Then, the semantic type of $\bar{\mathrm{C}}$ is going to be t as well. This implies that the relative pronoun must turn a truth-value into a function of type $\langle e, t\rangle$. Let us see how we can do this.

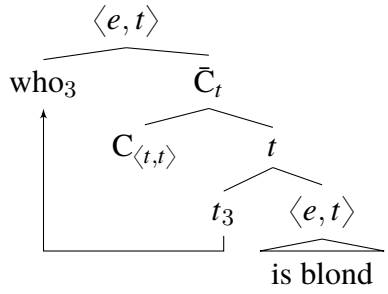
4 Relative Pronouns

In order to achieve what we want, we introduce a new compositional rule that is triggered by relative pronouns. It is called *Predicate Abstraction* (following Heim and Kratzer 1998).

(18) *Predicate Abstraction*: for any model M, for any assignment function a,
if RP is a relative pronoun with an index $i \in \mathbb{N}$,
then $\left[\!\left[\begin{array}{c} \mathrm{RP}_i \quad \mathrm{A} \\ \triangle \end{array} \right]\!\right]^{a,M} = \left[\lambda x \in D_e.\ \left[\!\left[\begin{array}{c} \mathrm{A} \\ \triangle \end{array} \right]\!\right]^{a[i \to x],M} \right]$

This rule is very complex. Let's unpack it step by step. Firstly, it says that whenever you see a relative pronoun, you use this rule. In other words, the rule is triggered by relative pronouns. In fact, relative pronouns themselves do not have denotations. After they trigger this rule, they sort of disappear, as you can see on the right-hand side of the equation. They are just triggers of the rule.

What does the rule do? It tells you how to interpret the sister constituent of the relative pronoun. The sister constituent is labeled A here, but it is always $\bar{\mathrm{C}}$, as ensured by the syntax. If you look at the right-hand side of '=' in (18), you see $\lambda x \in D_e$. So this rule adds the lambda term $\lambda x \in D_e$. Because of this, the result of the rule always has a type that looks like $\langle e, \tau\rangle$. Concretely, consider the following diagram.

(19)


As discussed in the previous section, $\bar{\mathrm{C}}$ is of type t. And Predicate Abstraction—triggered by the relative pronoun 'who$_3$'—adds '$\lambda x \in D_e$', which amounts to adding an extra type-e argument, so the entire relative clause is of type $\langle e, t\rangle$. This is exactly the type we want for the relative clause. It will combine with the noun it modifies via Predicate Modification, just like in the case of adjectives like 'blond'.

Adding '$\lambda x \in D_e$' is not the only thing that Predicate Abstraction does, however. Notice that the assignment function looks different on the right-hand side of '=' in (18), namely $a[i \to x]$ than on the left-hand side. This involves **assignment modification**. In words, $a[i \to x]$ is the assignment you obtain from a by minimally modifying it so that the index i is mapped to x.[2] Technically, it is defined as (20).

[2]Heim & Kratzer (1998) write $a^{x/i}$ for the same thing. I find this notation confusing. It is also harder to type as a itself is already a superscript on $[\![\ ]\!]$.

(20) For any assignment function a, for any index $i \in \mathbb{N}$ and for any individual x,

a. $a[i \to x](i) = x$; and

b. $a[i \to x](j) = a(j)$, for any index $j \in \mathbb{N}$ distinct from i.

The first clause (20a) says the modified assignment function $a[i \to x]$ maps i to x, and the second clause (20b) says for every other index j, the modified assignment function $a[i \to x]$ behaves just like the original assignment function a.

Here are some concrete examples. Suppose that g is the following assignment function:

$$g = \begin{bmatrix} 1 & \to & \text{Mary} \\ 2 & \to & \text{Tokyo} \\ 3 & \to & \text{London} \\ 4 & \to & \text{Kate} \\ 5 & \to & \text{Hanna} \\ \dots & & \end{bmatrix}$$

Let us modify this assignment function so that 3 will be mapped to NY instead of London. This modified assignment is denoted by $g[3 \to \text{NY}]$.

$$g[3 \to \text{NY}] = \begin{bmatrix} 1 & \to & \text{Mary} \\ 2 & \to & \text{Tokyo} \\ 3 & \to & \text{NY} \\ 4 & \to & \text{Kate} \\ 5 & \to & \text{Hanna} \\ \dots & & \end{bmatrix}$$

Notice that 3 is the only index that has been affected. Everything else stays intact.

Keep in mind that $a[i \to x]$ is an assignment function that might be different from a (though could be the same, e.g. $g = g[3 \to \text{London}]$). In particular, $a[i \to x]$ does *not* mean "a maps i to x". Typically this isn't the case. For example, if you consider $g[3 \to \text{NY}]$, $g(3) = \textit{London}$. Rather it means $g[3 \to \text{NY}](3) = \text{NY}$, and in every other respects, $g[3 \to \text{NY}]$ is identical to g.

Now we have all the ingredients necessary to derive the following result.

$$\begin{aligned} [\![\text{boy who}_3 \text{ C } t_3 \text{ is blond}]\!]^{a,M} &= [\lambda x \in D_e.\ 1 \text{ iff } x \text{ is blond } M \text{ and } x \text{ is a boy in } M] \\ &= [\![\text{blond boy}]\!]^{a,M} \end{aligned}$$

The following computation demonstrates this. Let us start with the denotation of the relative clause.

$$\left[\!\!\left[\begin{array}{c} \text{who}_3 \\ \text{C} \\ t_3 \\ \text{is} \quad \text{blond} \end{array} \right]\!\!\right]^{a,M}$$

$$= \left[\lambda x \in D_e. \left[\!\!\left[\begin{array}{c} \text{C} \\ t_3 \\ \text{is} \quad \text{blond} \end{array} \right]\!\!\right]^{a[3 \to x],M} \right] \qquad \text{(PA)}$$

$$= \left[\lambda x \in D_e.\ [\![\text{C}]\!]^{a[3\to x],M} \left(\left[\!\!\left[\begin{array}{c} t_3 \\ \text{is}\quad \text{blond} \end{array} \right]\!\!\right]^{a[3\to x],M} \right) \right] \qquad \text{(FA)}$$

$$= \left[\lambda x \in D_e.\ [\lambda v \in D_t.\ v] \left(\left[\!\!\left[\begin{array}{c} t_3 \\ \text{is}\quad \text{blond} \end{array} \right]\!\!\right]^{a[3\to x],M} \right) \right] \qquad \text{(Lexicon)}$$

$$= \left[\lambda x \in D_e.\ \left[\!\!\left[\begin{array}{c} t_3 \\ \text{is}\quad \text{blond} \end{array} \right]\!\!\right]^{a[3\to x],M} \right] \qquad (\lambda\text{-conv.})$$

$$= \left[\lambda x \in D_e.\ \left[\!\!\left[\begin{array}{c} \text{is}\quad \text{blond} \end{array} \right]\!\!\right]^{a[3\to x],M} \left([\![t_3]\!]^{a[3\to x],M} \right) \right] \qquad \text{(FA)}$$

$$= \left[\lambda x \in D_e.\ \left[\!\!\left[\begin{array}{c} \text{is}\quad \text{blond} \end{array} \right]\!\!\right]^{a[3\to x],M} (a[3 \to x](3)) \right] \qquad \text{(TR)}$$

$$= \left[\lambda x \in D_e.\ \left[\!\!\left[\begin{array}{c} \text{is}\quad \text{blond} \end{array} \right]\!\!\right]^{a[3\to x],M} (x) \right]$$

$$= \left[\lambda x \in D_e.\ [[\text{is}]]^{a[3\to x],M}([[\text{blond}]]^{a[3\to x],M})](x) \right] \qquad \text{(FA)}$$

$$= \left[\lambda x \in D_e.\ [[\lambda f \in D_{\langle e,t\rangle}.\ f]([[\text{blond}]]^{a[3\to x],M})](x) \right] \qquad \text{(Lexicon)}$$

$$= \left[\lambda x \in D_e.\ [[\text{blond}]]^{a[3\to x],M}(x) \right] \qquad (\lambda\text{-conv.})$$

$$= [\lambda x \in D_e.\ [\lambda y \in D_e.1 \text{ iff } y \text{ is blond in } M](x)] \qquad \text{(Lexicon)}$$

$$= [\lambda x \in D_e.\ 1 \text{ iff } x \text{ is blond in } M] \qquad (\lambda\text{-conv.})$$

This is exactly the same denotation as $[\![\text{blond}]\!]^{a,M}$. Now we can combine this with $[\![\text{boy}]\!]^{a,M}$ via Predicate Modification:

$$\left[\!\!\left[\begin{array}{c} \text{boy} \\ \text{who}_3 \\ \text{C} \\ t_3 \\ \text{is}\quad \text{blond} \end{array} \right]\!\!\right]^{a,M}$$

$$= \left[\lambda y \in D_e.\ [[\text{boy}]]^{a,M}(y) = 1 \text{ and } \left[\!\!\left[\begin{array}{c} \text{who}_3 \\ \text{C} \\ t_3 \\ \text{is}\quad \text{blond} \end{array} \right]\!\!\right]^{a,M} (y) = 1 \right] \qquad \text{(PM)}$$

$$= [\lambda y \in D_e.\ [[\text{boy}]]^{a,M}(y) = 1 \text{ and } [\lambda x \in D_e.\ 1 \text{ iff } x \text{ is blond in } M](y) = 1] \qquad \text{(see above)}$$

$= [\lambda y \in D_e.\ [\![\text{boy}]\!]^{a,M}(y) = 1 \text{ and } y \text{ is blond in } M]$ (λ-conv.)

$= [\lambda y \in D_e.\ [\lambda x \in D_e.\ 1 \text{ iff } x \text{ is a boy in } M](y) = 1 \text{ and } y \text{ is blond in } M]$ (Lexicon)

$= [\lambda y \in D_e.\ y \text{ is a boy in } M \text{ and } y \text{ is blond in } M]$ (λ-conv.)

Therefore, we have: $[\![\text{blond boy}]\!]^{a,M} = [\![\text{boy who}_3 \text{ C } t_3 \text{ is blond}]\!]^{a,M}$.

There are several remarks to make. In performing rules like Functional Application and Predicate Modification, you have to carry over the same assignment function as the mother node. Concretely, in the computation of the denotation of the relative clause above, the second line performs Predicate Abstraction and modifies the assignment function to $a[3 \to x]$. In the next line, FA decomposes 'C t_3 is blond' into 'C' and 't_3 is blond'. Both of these components are interpreted with respect to the modified assignment function $a[3 \to x]$. You do not suddenly go back to a here. Similarly, in line 6, Functional Application applies to 't_3 is blond', splitting it into 't_3' and 'is blond'. Here it is crucial that both of these components are evaluated with respect to the modified assignment function $a[3 \to x]$, because the denotation of the trace is dependent on the assignment function. If we used the unmodified assignment function a instead, we wouldn't get x! And notice that this is exactly the mechanism that achieves the binding of the trace with λx.

If this is the first time you read this, you probably didn't understand the intricate and complex interplay between Predicate Abstraction and traces. This is quite natural. You have to go through actual computations yourself to really understand what's going on. To this end, you will do a computation in the homework assignment for this week. We will also discuss this in class.

Before wrapping up, there are several technical loose ends to tie up, although you do not need to understand the details the first time you read this. That is, to ensure that assignment modification percolates down as in the above computation, we need to re-state the compositional rules by adding 'for any assignment function a' (The previous versions did not mention assignment functions).

(21) *Functional Application*: For any model M and for any assignment function a, if A is a branching node with B and C as its daughters and $[\![\text{B}]\!]^{a,M}$ is of type $\langle\sigma, \tau\rangle$ and $[\![\text{C}]\!]^{a,M}$ is of type σ, then $[\![\text{A}]\!]^{a,M} = [\![\text{B}]\!]^{a,M}([\![\text{C}]\!]^{a,M})$.

(22) *Predicate Modification*: For any model M and for any assignment function a, if A is a branching node with B and C as its daughters and $[\![\text{B}]\!]^{a,M}$ and $[\![\text{C}]\!]^{a,M}$ are both of type $\langle e, t\rangle$, then $[\![\text{A}]\!]^{a,M} = [\lambda x \in D_e.\ [\![\text{B}]\!]^{a,M}(x) = 1 \text{ and } [\![\text{C}]\!]^{a,M}(x) = 1]$.

(23) *Non-Branching Node Rule*: For any model M and for any assignment function a, If A is a non-branching node with B as its sole daughter, then $[\![\text{A}]\!]^{a,M} = [\![\text{B}]\!]^{a,M}$.

When we say 'for any assignment function a', modified assignment functions are included. Modified assignment functions are assignment functions by definition.

Finally, it should also be noted that when Predicate Abstraction is involved, the computation is much easier when done top-down, as in the above examples, because if you go bottom-up, whenever you interpret a trace, you need to look ahead and note that the assignment function will be modified later on. Here is a concrete example. To save space, I will not represent the structure in this example.

- By Trace Rule, $[\![t_3]\!]^{a,M} = a(3)$. (Note that a here is actually a modified assignment function, but we cannot know how it will be modified at this point.)
- (Skipping some steps) By FA $[\![t_3 \text{ is blond}]\!]^{a,M} = [\![\text{is blond}]\!]^{a,M}(a(3)) = 1$ iff $a(3)$ is blond in M.

- (Again skipping some steps) By FA, $[\![\text{C } t_3 \text{ is blond}]\!]^{a,M} = [\![t_3 \text{ is blond}]\!]^{a,M} = 1$ iff $a(3)$ is blond in M.
- Finally, by PA, $[\![\text{who}_3 \text{ C } t_3 \text{ is blond}]\!]^{b,M} = [\lambda x \in D_e.[\![\text{C } t_3 \text{ is blond}]\!]^{a,M}]$ where $b[3 \to x] = a$.

In the last step, I needed to name the assignment for the whole relative clause b, because it's a different assignment function from a. In fact, you get a by modifying b, i.e. $b[3 \to x] = a$. This computation is not wrong, but is harder to understand. So from now on, do top-down computation.

5 Summary

We have introduced a lot of new things. First, the denotations of traces are dependent on assignment functions, as stated in a new compositional rule, *Trace Rule*.

(24) *Trace Rule*: For any model M and for any assignment function a and for any index $i \in \mathbb{N}$, $[\![t_i]\!]^{a,M} = a(i)$.

Assignment functions are simply functions from indices (=natural numbers) to individuals.

Their primary role is to be bound by a λ-operator introduced by a relative pronoun (of which it is a trace). This is achieved by another new compositional rule, *Predicate Abstraction*.

(25) *Predicate Abstraction*: for any model M, for any assignment function a,
if RP is a relative pronoun with an index $i \in \mathbb{N}$,
then $\left[\!\left[\begin{array}{c} \text{RP}_i \quad \text{A} \end{array} \right]\!\right]^{a,M} = \left[\lambda x \in D_e. \left[\!\left[\begin{array}{c} \text{A} \end{array} \right]\!\right]^{a[i \to x],M} \right]$

$a[i \to x]$ is a modified assignment function, which is just like the original assignment function a except $a[i \to x](i) = x$.

(26) For any assignment function a, for any index $i \in \mathbb{N}$ and for any individual x,
a. $a[i \to x](i) = x$; and
b. $a[i \to x](j) = a(j)$, for any index $j \in \mathbb{N}$ distinct from i.

Finally, the old compositional rules now also refer to an assignment function. They make sure that once an assignment function is modified, all the constituents below that point will be interpreted relative to the modified assignment function.

(27) *Functional Application*: For any model M and for any assignment function a,
if A is a branching node with B and C as its daughters and $[\![\text{B}]\!]^{a,M}$ is of type $\langle \sigma, \tau \rangle$ and $[\![\text{C}]\!]^{a,M}$ is of type σ, then $[\![\text{A}]\!]^{a,M} = [\![\text{B}]\!]^{a,M}([\![\text{C}]\!]^{a,M})$.

(28) *Predicate Modification*: For any model M and for any assignment function a,
if A is a branching node with B and C as its daughters and $[\![\text{B}]\!]^{a,M}$ and $[\![\text{C}]\!]^{a,M}$ are both of type $\langle e, t \rangle$, then $[\![\text{A}]\!]^{a,M} = [\lambda x \in D_e.\ [\![\text{B}]\!]^{a,M}(x) = 1 \text{ and } [\![\text{C}]\!]^{a,M}(x) = 1]$.

(29) *Non-Branching Node Rule*: For any model M and for any assignment function a,
If A is a non-branching node with B as its sole daughter, then $[\![\text{A}]\!]^{a,M} = [\![\text{B}]\!]^{a,M}$.

6 Other Types of Relative Clauses

We now have an analysis of *wh*-relatives in Englsih.

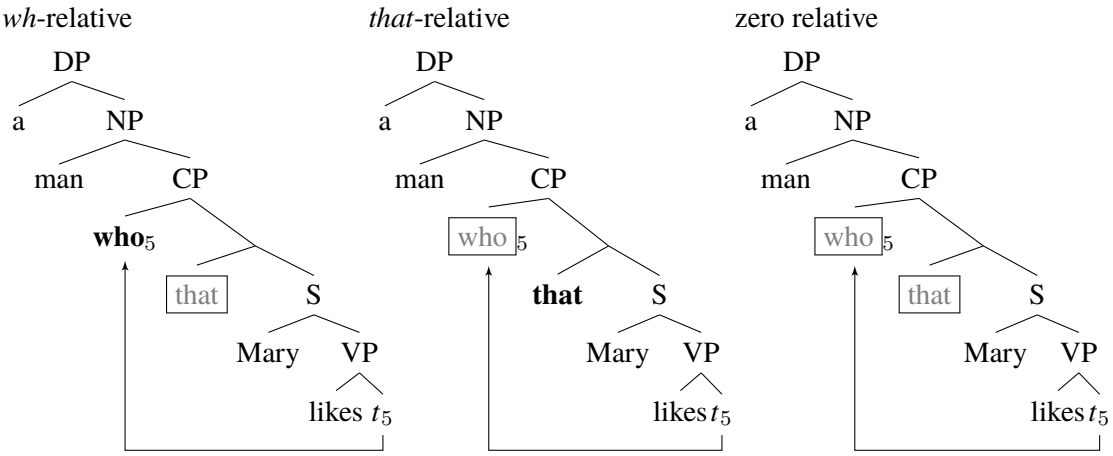
(30) John is a man who Mary likes.

What about other types of relative clauses we saw at the beginning?

(31) a. John is a man that Mary likes (*that*-relative)
b. John is a man Mary likes (zero relative)

We can actually use the same mechanisms to analyse these sentences (thank God!), if we assume that they have the same structure as (30). Recall in particular that by assumption, (30) involves a null complementizer C. We can analyze (31a) as minimally different from (30) in that it is the complementizer that is overt here and the relative pronoun is phonologically null. Furthermore, for (31b), we can say that both the complementizer and the relative pronoun are left unpronounced. Then, all of these sentences have the exact same structure, as depicted in the following diagrams, and they only differ in which component in CP is pronounced. The phonologically null elements are marked by $\boxed{}$.

(32)



Interestingly, it is ungrammatical to pronounce both the relative pronoun 'who' and the complementizer 'that'.

(33) *John is a man who that Mary likes.

So in (modern) English, at most one element in CP can be pronounced. We do not mean to discuss why this is so here, as this seems to be a syntactic issue, but as a final note, it should be mentioned that in some languages, both the relative pronoun and the complementizer can be pronounced. The following data are from Santorini & Kroch (2007; *The Syntax of Natural Language*).

(34) thy freend **which that** thou has lorn.
your friend which that you have lost
'your friend that you have lost' (Middle English; *The Canterbury Tales*)

(35) der Hund **der wo** gestern d' Katz bissn hod.
the dog who that yesterday the cat bitten has
'the dog that bit the cat yesterday' (Bavarian)