5 Relative Clauses, Variables, Variable Binding

In this chapter, we consider the internal semantics of another kind of NP-modifier, the relative clause. This will give us occasion to introduce variables and variable binding. The standard interpretation techniques for structures with variables work with variable assignments. Since the notion of a "variable assignment" is not an easy one, we will introduce it in a stepwise fashion. We will start out with a simplified notion of a variable assignment that allows us to interpret structures with (possibly multiple occurrences) of just one variable. Once the essential methods of variable interpretation and variable binding are in place, we will introduce the general notion of a variable assignment and look at structures with multiple variables. The final two sections of the chapter are devoted to general issues of variable binding in syntax and semantics.

5.1 Relative clauses as predicates

Our analysis of relative clauses goes back at least to Quine:[1]

> The use of the word "relative" in "relative clause" has little to do with its use in "relative term".[2] A relative clause is usually an absolute term. It has the form of a sentence except that a relative pronoun stands in it where a singular term[3] would be needed to make a sentence, and often the word order is switched; thus "which I bought". *A general term of this sort is true of just those things which, if named in the place of the relative pronoun, would yield a true sentence; thus "which I bought" is true of just those things x such that x I bought, or, better, such that I bought x.*
>
> From this last broad rule we see in particular that a relative pronoun is in a way redundant when it occurs as subject. For example, "who loves Mabel" is true of just the persons of whom "loves Mabel" is true, and "which is bigger than Roxbury" is true of just the things of which "bigger than Roxbury" is true. But the redundant pronoun can serve a grammatical

purpose: we switch from "loves Mabel" to "who loves Mabel" for attributive use as in "brother who loves Mabel", just because relative clauses are adjectival[4] and hence suited, unlike the verbal form "loves Mabel", to attributive position. There is less purpose in "which is bigger than Roxbury", since "bigger than Roxbury" is adjectival already. The main use of a form like "which is bigger than Roxbury" is after a comma as an unrestrictive clause; and we may pass over unrestrictive clauses, for they are only stylistic variants of coordinate sentences.

At any rate *the peculiar genius of the relative clause is that it creates from a sentence "... x ..." a complex adjective summing up what that sentence says about x*. Sometimes the same effect could be got by dropping "x is", as in the last example, or by other expedients; thus, in the case of "I bought x", "bought by me" (formed by conversion and application[5]) would serve as well as the relative clause "which I bought". But often, as in the case of "the bell tolls for x", the relative clause is the most concise adjective available for the purpose.

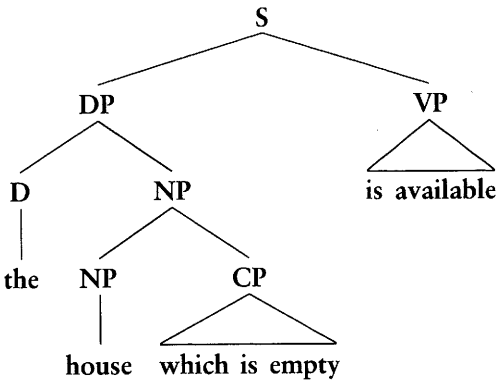
... A fruitful basis for singular descriptions is the general term of the form of a relative clause; thence "the car [which] I bought from you". Let us build this example from its elements. We have a triadic relative term "bought from", which, applied predicatively to the singular terms "I", "x" (say), and "you", gives a sentence form "I bought x from you". Putting the relative pronoun for the "x" here and permuting, we get the relative clause "which I bought from you". This clause is a general term, adjectival in status. Combining it attributively with the general term "car", we get the general term "car which I bought from you"; and then "the" yields the singular term.

As Quine makes clear, if we abstract away from their internal syntactic and semantic composition, relative clauses are just like other modifiers in NP – for example, the PPs and APs we considered earlier. They have the same type of denotation (namely, characteristic functions of sets), and they contribute in the same way to the denotation of the surrounding structure. The latter actually *follows* from the former in our framework. Since we don't allow construction-specific semantic rules but only general principles of composition, we are committed to the prediction that phrases in the same environment and with the same type of denotation must make the same contribution. Let's consider an example:

(1) **The house which is empty is available.**

Omitting the internal structure of the relative clause, the determiner phrase (DP) in (1) has a structure of the following kind:

(1′)



"House", "empty", and "available" have the obvious lexical entries; for example, ⟦**empty**⟧ = $\lambda x \in D_e$. x is empty. We also have a denotation for the determiner "the". So once we decide on an interpretation for the complementizer phrase (CP) "which is empty", we can calculate the truth-conditions of (1′).

Following Quine, we hypothesize that "which is empty" has exactly the same denotation as "empty". This is suggested by the fact that substituting "empty" for "which is empty" leads to a sentence with equivalent truth-conditions: "The empty house is available". (Never mind the change in order, which is for some reason required by the syntax of English.) The rest now follows: ⟦**house**⟧ and ⟦**which is empty**⟧ combine by Predicate Modification (PM), with the result that the NP "house which is empty" denotes the function $\lambda x \in D_e$. x is a house and x is empty. When we apply Functional Application (FA) to compute the denotation of the DP above, this function becomes the argument of ⟦**the**⟧. According to the entry for "the", then, the DP "the house which is empty" has a denotation iff there is exactly one empty house, and in that case it denotes the unique empty house. Finally, the whole S (again by FA) has a truth-value iff there is a unique empty house, and it has the value 1 iff the unique empty house is available. This prediction conforms to intuitive judgment.

Notice that this analysis correctly distinguishes restrictive relatives from their nonrestrictive counterparts: for example, (1) from (2).

(2) **The house, which is empty, is available.**

If we assume (as we did above and as does Quine) that nonrestrictive modifiers are like separate sentences, then ⟦**the**⟧ in (2) applies to the extension of "house" by itself (and not to the extension of any constituent including "which is empty"). This implies that (2) presupposes there to be exactly one house – unlike (1), which is entirely compatible with there being two or more houses, as long as only one of them is empty.[6] In what follows, we will disregard nonrestrictives and talk exclusively about restrictive relatives.

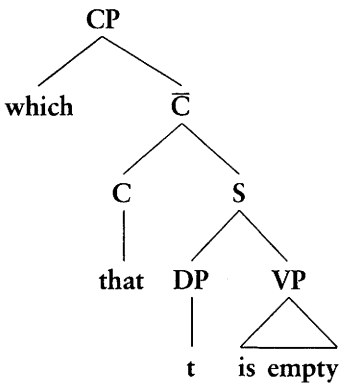
To sum up, restrictive relatives are just another kind of intersective modifier.

5.2 Semantic composition inside the relative clause

What remains to be worked out, then, is the internal semantic composition of relative clauses. What are their internal syntactic structures, and what lexical entries and composition principles are needed to make them denote the appropriate functions of type <e,t>? For instance, how do we derive systematically what we simply assumed above: namely, that **[[which is empty]]** = **[[empty]]**?

We will adopt a more or less standard syntactic analysis of relative clauses, according to which they look like this:

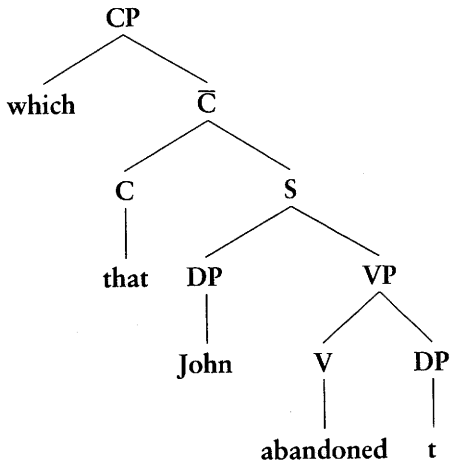
(1)



Various other structures would serve our purposes equally well, as long as there is a relative pronoun at the top and a trace in subject position. In (1), either the complementizer (C) or the *wh*-word has to be deleted on the surface.[7] We categorized the trace in (1) as a DP (determiner phrase). From now on, we will take all phrases that show the same syntactic behavior as phrases headed by overt determiners to be DPs. This includes proper names, pronouns, and traces.[8]

The semantics for structures like (1) will require some innovative additions to our current theory. What we know at this point is what we want to come out on top: The CP (complementizer phrase) should get a value in $D_{<e,t>}$; more particularly, in the case of (1), it should denote the characteristic function of the set of empty objects. We also know what the VP inside this CP denotes; it's actually that very same function. But it would be too easy to assume therefore that the semantic value simply gets passed up from VP to S to $\bar{C}$ to CP. That would work in this special case, but not (as Quine already remarked) in general. We must also worry about cases where the trace is not in subject position:

(2)



(2) should denote the function $\lambda x \in D$. John abandoned x. In this case, this is not the value of any of its subtrees.

The basic question we face at this point is: What are the semantic values of traces? For instance, what is the extension of the object DP in (2)?

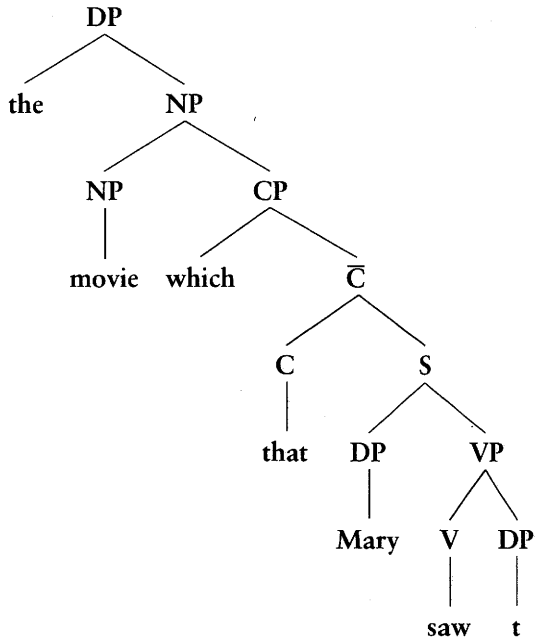
We face a dilemma here. On the one hand, we would like traces to have the same type of extension as other DPs, because then we could use the same composition principles as before to interpret the structures they appear in. For instance, we would like "t" in (2) to denote an individual, because that would make a suitable argument for the extension of the verb "abandon". If we treated the trace as semantically vacuous, for instance, we would be in trouble: the S-node would denote the characteristic function of the set of all individuals who abandoned John – with John being the one abandoned instead of the abandoner! On the other hand, does it make sense to assign a referent to the trace? Let's investigate this possibility.

5.2.1 Does the trace pick up a referent?

It is sometimes suggested that the relative pronoun is anaphorically related to and inherits the reference of the relative clause's head.[9] For instance, in "the movie (which) Mary saw t", the trace is said to get its denotation from "which", which in turn gets it from the head. One immediate objection to this idea is that it would not apply to relatives in quantifier phrases like "no movie that Mary saw", since such phrases do not denote individuals (as we will argue in more detail below). But even if we stick to definite DPs, it is an idea that raises puzzling questions.

First, what do we mean by the "head" of the relative clause? In the syntactic structure we are assuming, this could only be the whole containing DP:

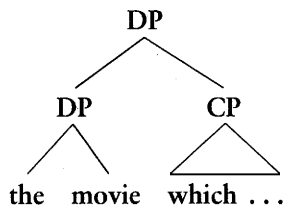
(3)



There just is no smaller DP in this structure. In fact, there is no smaller constituent of *any* kind that denotes an individual and from which a referent of the desired type e could thus be picked up by the trace. The whole DP does denote an individual. But if we said that the trace inherits its denotation from that, we would get into a vicious circle: The denotation of the whole is supposed to be built up from the denotations of its parts, and this construction can't get off the ground if we need the denotation of the whole before we can interpret some of the parts.

Should we reconsider the structure, then, and adopt the following bracketing after all?

(4)



No, that just gets us into different problems.[10] Now ⟦**the**⟧ applies to ⟦**movie**⟧, giving rise to a presupposition that there is just one (relevant) movie. Besides, it is mysterious what the CP would have to denote. It couldn't be a function of type <e,t>, because then the whole DP would denote (if anything) a truth-value instead of an individual.

Let us abandon this line of approach. Here, then, is the dilemma: We would like the trace to denote an individual so that we can interpret the nodes above it, but we can't seem to find a suitable individual. There is no easy way out within the confines of our current theoretical apparatus. It is time to explore the utility of a genuinely new theoretical construct, the *variable*.

5.2.2 *Variables*

Variables were invented precisely to be like ordinary referring phrases in the respects we want them to be, but sufficiently unlike them to avoid the puzzles we just ran up against. A variable denotes an individual, but only *relative to a choice of an assignment of a value*. What is a value assignment for a variable? The simplest definition for our present purposes is this:

(5) Preliminary definition: An *assignment* is an individual (that is, an element of D (= D_e)).

A trace under a given assignment denotes the individual that constitutes that assignment; for example:

(6) The denotation of "t" under the assignment Texas is Texas.

An appropriate notation to abbreviate such statements needs to be a little more elaborate than the simple ⟦ . . . ⟧ brackets we have used up to now. We will indicate the assignment as a superscript on the brackets; for instance, (7) will abbreviate (6):

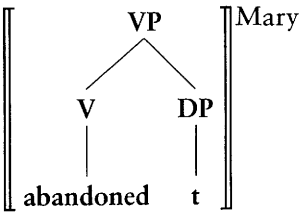
(7) $[\![t]\!]^{Texas}$ = Texas.

The general convention for reading this notation is as follows: Read "$[\![\alpha]\!]^a$" as "the denotation of α under a" (where α is a tree and a is an assignment).

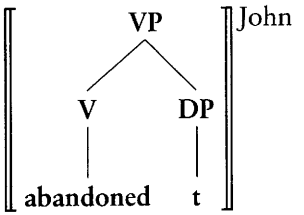
(7) exemplifies a special case of a general rule for the interpretation of traces, which we can formulate as follows:

(8) If α is a trace, then, for any assignment a, $[\![\alpha]\!]^a$ = a.

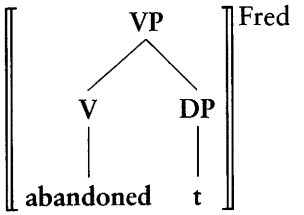
The decision to relativize the denotations of traces to assignments has repercussions throughout our system of rules. We must allow the denotations of larger phrases that contain traces to be assignment-relative as well. For instance, a VP whose object is a trace will not denote a fixed function in $D_{<e,t>}$, but may denote different functions under different assignment functions; for instance:



$= \lambda x \in D_e$. x abandoned Mary

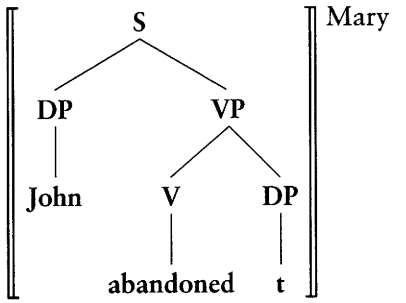


$= \lambda x \in D_e$. x abandoned John

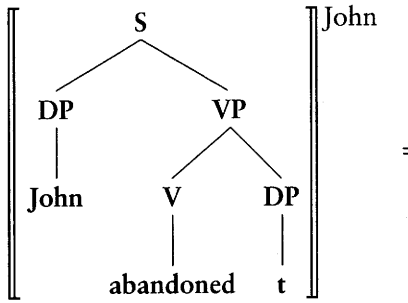


$= \lambda x \in D_e$. x abandoned Fred

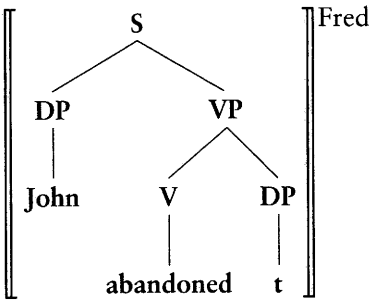
A sentence like **John abandoned t**, then, does not have truth-conditions *per se*, but only with respect to an assignment. We have, for example:



= 1 iff John abandoned Mary



= 1 iff John abandoned John



= 1 iff John abandoned Fred

Now that we have admitted traces into our syntactic representations, we find ourselves in a rather odd situation. We have sentences like "John abandoned Mary" that have truth-conditions (and hence a meaning) *per se*, and our theory should acknowledge this fact, as it has always done. But we also have to deal with sentences like "John abandoned t" that may appear as parts of relative clauses, and need to have assignment-dependent denotations. How can we do justice to both types of sentences without complicating our composition principles? Take the Functional Application principle, for instance. It should be written in such a way that the top node gets an assignment-dependent value whenever either of the daughters does. Now it would be rather inelegant to have to distinguish three different cases here, according to whether the function-denoting daughter, or the argument-daughter, or neither of them happens to contain a trace and therefore to have an assignment-relative value. It is simpler, if a bit artificial, to formulate all our composition principles for assignment-dependent denotations and introduce assignment-independent denotations through a definition, as follows:

(9) For any tree α, α is in the domain of $[\![\ ]\!]$ iff for all assignments a and b, $[\![\alpha]\!]^a = [\![\alpha]\!]^b$.
If α is in the domain of $[\![\ ]\!]$, then for all assignments a, $[\![\alpha]\!] = [\![\alpha]\!]^a$.

As for lexical rules, if we have an item like **laugh**, we can still assign it the assignment-independent denotation $[\![\textbf{laugh}]\!]$. But with definition (9), we automatically also have a semantic value for this item under any arbitrary assignment. From (9) and the lexical entry for **laugh**, it follows that

(10) For any assignment a, $[\![\textbf{laugh}]\!]^a = [\![\textbf{laugh}]\!] = \lambda x \in D_e$. x laughs.

Now we can write the new versions of our composition principles almost as simply as the old ones, except that we need to distinguish two cases in the interpretation of terminal nodes. Our previous Terminal Nodes rule now divides into (8) above and (11).[11]

(11) *Lexical Terminals*
If α is a terminal node occupied by a lexical item, then $[\![\alpha]\!]$ is specified in the lexicon.

(12) *Non-Branching Nodes* (NN)
If α is a non-branching node and β its daughter, then, for any assignment a, $[\![\alpha]\!]^a = [\![\beta]\!]^a$.

(13) *Functional Application* (FA)
If α is a branching node and {β, γ} the set of its daughters, then, for any assignment a, if $[\![\beta]\!]^a$ is a function whose domain contains $[\![\gamma]\!]^a$, then $[\![\alpha]\!]^a = [\![\beta]\!]^a([\![\gamma]\!]^a)$.

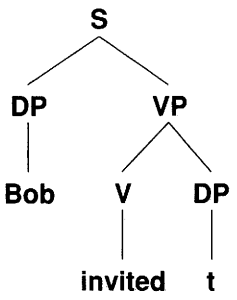
(14) *Predicate Modification* (PM)
If α is a branching node and {β, γ} the set of its daughters, then, for any assignment a, if $[\![\beta]\!]^a$ and $[\![\gamma]\!]^a$ are both functions of type <e,t>, then $[\![\alpha]\!]^a = \lambda x \in D\ .\ [\![\beta]\!]^a(x) = [\![\gamma]\!]^a(x) = 1$.

Exercise

Consider a state of affairs with the following properties:
D (= D_e) = {b, j, m, s}.
s invites b and m; b invites b, j, and m; no other invitations take place.
$[\![\textbf{Bob}]\!]$ = b.

(a) Show that $[\![(i)]\!]^m = 1$.

(i)



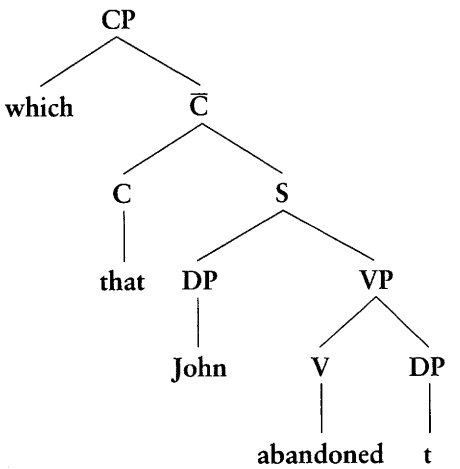
(b) Which assignments make (i) true? (List all.)

(c) Show that $[\![(i)]\!]$ is undefined: that is, that the tree under (i) is not in the domain of $[\![\ ]\!]$.

5.2.3 *Predicate abstraction*

Finally, we can start to think about the semantic principle that determines the denotation of a relative clause like (2) (repeated from above) from its two immediate constituents: the relative pronoun **which** and the sentence **John abandoned t.**

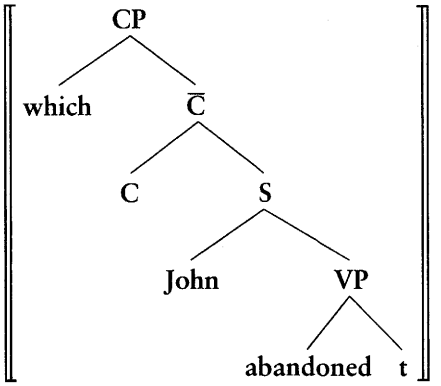
(2)



We treat the complementizer "that" as semantically vacuous, so the $\bar{C}$ inherits the value of the S below it. The relative pronoun within CP is also not assigned any denotation of its own. But it is not simply vacuous; its presence will be required to meet the structural description of the composition principle applying to the CP above it. That principle is a new one, unrelated to the ones we have employed thus far:[12]

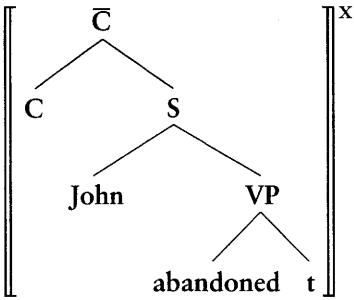
(15) *Predicate Abstraction* (PA)
If α is a branching node whose daughters are a relative pronoun and β, then $[\![\alpha]\!] = \lambda x \in D \,.\, [\![\beta]\!]^x$.

This rule is also known as "Functional Abstraction" or "Lambda Abstraction" in the literature. The reason for these names is transparent. The semantic value of α is defined as a function. In fact, our formulation of PA uses the λ-notation to define this function. (This was not, of course, essential; we could have defined it in words as well.) To appreciate what the rule is doing, look again at (2). Being a relative clause, (2) should have an assignment-independent interpretation, and it should denote the function $\lambda x \in D$. John abandoned x. Here is the proof that it does:



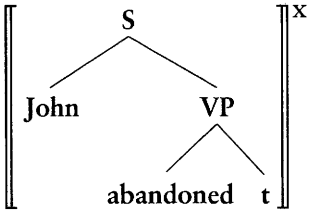
= (by PA)

$\lambda x \in D$.



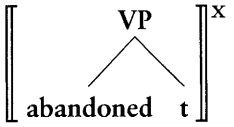
= (by vacuity of C)

$\lambda x \in D$.



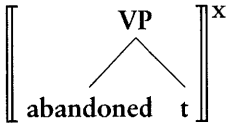
= (by FA)

$\lambda x \in D$.



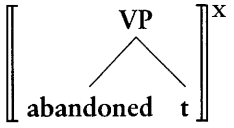
($[\![$**John**$]\!]^x$) = (by definition (9))

$\lambda x \in D$.



($[\![$**John**$]\!]$) = (by lexical entry of **John**)

$\lambda x \in D$.



(John) = (by FA)

$\lambda x \in D$. $[\![$**abandoned**$]\!]^x([\![t]\!]^x)$(John) = (by Traces Rule)

$\lambda x \in D$. $[\![$**abandoned**$]\!]^x(x)$(John) = (by definition (9))

$\lambda x \in D$. [**abandoned**](x)(John) = (by lexical entry of **abandoned**)

$\lambda x \in D$. [$\lambda y \in D$. [$\lambda z \in D$. z abandoned y]](x)(John)
= (by definition of λ-notation)

$\lambda x \in D$. John abandoned x. QED.

The Predicate Abstraction Rule gives the moved relative pronoun what is called a *syncategorematic* treatment. Syncategorematic items don't have semantic values of their own, but their presence affects the calculation of the semantic value for the next higher constituent. A syncategorematic treatment of relative pronouns goes against the concept of type-driven interpretation that we argued for earlier, and we will eventually want to abolish rules of this kind. At the moment, we will not worry about this blemish of our theory, however, and will keep the current version of the Predicate Abstraction Rule (at least for a little while), since it is easier to handle for beginning semanticists than the theoretically more adequate alternatives.

When you work with assignments, do not confuse denotations *under* assignments with denotations *applied to* assignments. There is a big difference between $[\![a]\!]^x$ and [a](x). For example,

[whom John abandoned t]$^{\text{Charles}} \neq$ **[whom John abandoned t]**(Charles)
[sleeps]$^{\text{Ann}} \neq$ **[sleeps]**(Ann)

What you see on the left side is a function of type <e,t>, but what you have on the right is the result of applying such a function to an individual – in other words, a truth-value. In many other instances, one of the two notations isn't even well-defined in the first place. For example,

"**[John abandoned t]**^{x}"

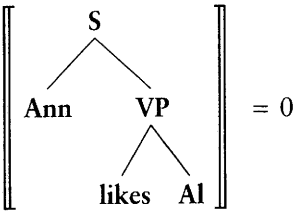
makes sense: it stands for a truth-value; that is, it equals either 1 or 0, depending on what individual "x" stands for.

"**[John abandoned t]**(x)"

on the other hand is nonsense, for two reasons. First, it falsely presumes that "John abandoned t" is in the domain of [ ]; that is, that it has a semantic value independently of a specified assignment. This is not the case. Second, even if "John abandoned t" did happen to have an assignment-independent denotation, its denotation (like that of any other S) would be a truth-value, not a function. So it would be as though we had written "1(x)" or "0(x)".

5.2.4 A note on proof strategy: bottom up or top down?

When you are asked to calculate the semantic value of a given tree, you have a choice between two strategies: to work from the bottom up or from the top down. To take a very simple example, you are told that, as a matter of actual fact, Ann doesn't like Al, and you are asked to show that

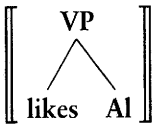


$= 0$

One way to present your proof is as follows:

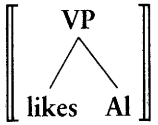
Bottom-up proof

By lexicon: $[\![\textbf{Ann}]\!]$ = Ann (=: (i))
By lexicon: $[\![\textbf{likes}]\!] = \lambda x \in D \,.\, \lambda y \in D \,.\, y$ likes x (=: (ii))
By lexicon: $[\![\textbf{Al}]\!]$ = Al (=: (iii))

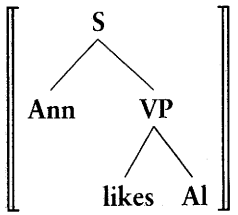
By FA: 

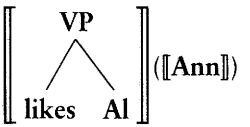
$= [\![\textbf{likes}]\!]([\![\textbf{Al}]\!])$

Therefore, using (ii) and (iii) from above:



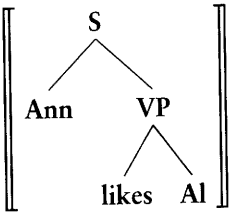
$= [\lambda x \in D \,.\, \lambda y \in D \,.\, y \text{ likes } x](\text{Al}) = \lambda y \in D \,.\, y$ likes Al (=: (iv))



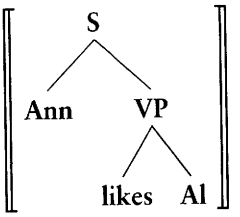
$=$ 

$([\![\textbf{Ann}]\!])$

Therefore, using (i) and (iv) from above:

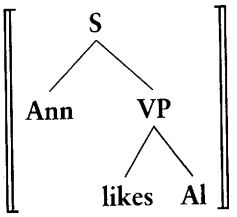


$= [\lambda y \in D \; . \; y \text{ likes Al}](\text{Ann})$

This means: 

= 1 iff Ann likes Al.

And given the description of the facts, Ann doesn't like Al, therefore

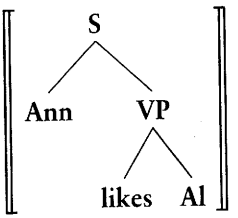


= 0.

QED.

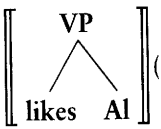
This was the bottom-up strategy, because we calculated values for the lowest nodes first and got to the top node last. Alternatively, you could have presented your reasoning as follows:

Top-down proof



= 1

iff (by FA)



([[Ann]]) = 1

iff (by FA)

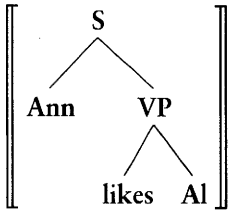
$[\![\textbf{likes}]\!]([\![\textbf{Al}]\!])([\![\textbf{Ann}]\!]) = 1$

iff (by lexical entries)

$[\lambda x \in D \,.\, \lambda y \in D \,.\, y \text{ likes } x](\text{Al})(\text{Ann}) = 1$

iff (by definition of λ-notation)

Ann likes Al.

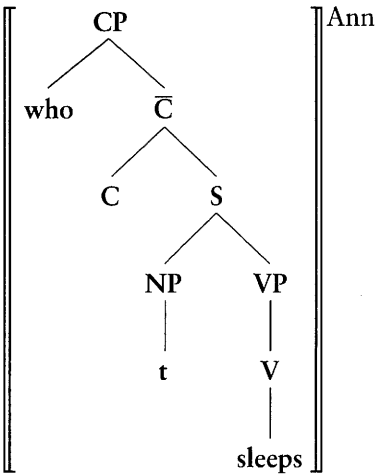
Since she doesn't, therefore, 

$= 0.$

QED.

This second strategy was the top-down strategy. Both strategies are equally sound, and there is no major difference in efficiency. Most people are naturally inclined towards the bottom-up strategy at first, but there is really no reason to prefer it.

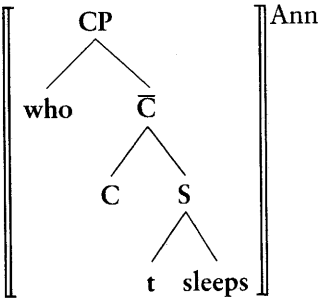
As we turn to calculations involving Predicate Abstraction, however, the top-down method begins to have an advantage. Suppose your assignment is to show that the following equation holds:



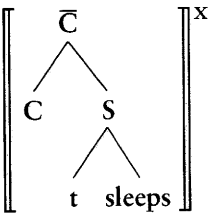
$= \lambda x \in D \,.\, x \text{ sleeps}.$

Let's first prove this by the top-down method.

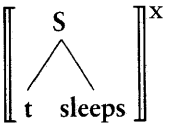
Top-down proof



= (by PA and definition (9))

$\lambda x \in D$. 

= (by vacuity of C)

$\lambda x \in D$. 

= (by FA)

$\lambda x \in D$. $[[\textbf{sleep}]^x([[\textbf{t}]^x)]$ = (by Traces Rule)

$\lambda x \in D$. $[[\textbf{sleep}]^x(x)]$ = (by lexical entry of **sleep** and definition (9))

$\lambda x \in D$. $[[\lambda y \in D$. y sleeps$](x)]$ = (by definition of λ-notation)

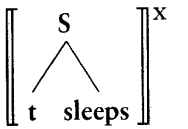
$\lambda x \in D$. x sleeps.

QED.

This was straightforward. Now suppose we had tried to do it bottom up. Here is the right way to do that:

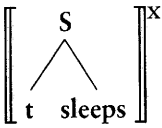
Bottom-up proof

Let $x \in D$ $(= D_e)$ be an arbitrarily chosen assignment.
By the rule for traces: $[[\textbf{t}]]^x = x$ (=: (i))
$[[\textbf{sleeps}]]^x = \lambda y \in D$. y sleeps. [=: (ii)]

By FA: 

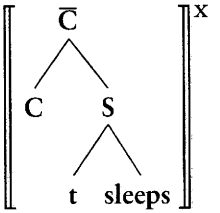
$= [[\textbf{sleeps}]]^x([[\textbf{t}]]^x)$

Therefore, using (i) and (ii) from above:



$= [\lambda y \in D \,.\, y \text{ sleeps}](x).$

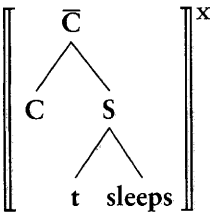
By vacuity of C:



$= [\lambda y \in D \,.\, y \text{ sleeps}](x)$

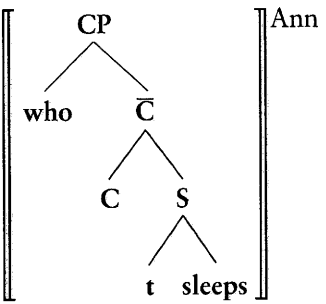
Since x was arbitrary, we can now summarize:

For any $x \in D$:

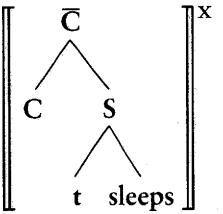


$= [\lambda y \in D \,.\, y \text{ sleeps}](x)$ [=: (iii)]

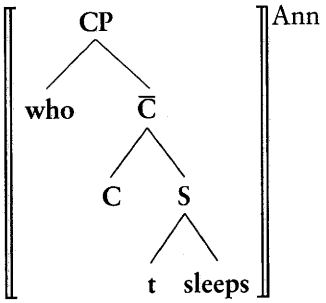
By PA and definition (9):



$= \lambda x \in D \,.$

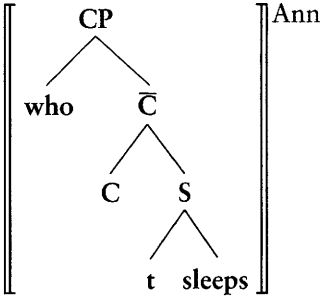


By (iii) above, this means:



$= \lambda x \in D \,.\, [[\lambda y \in D \,.\, y \text{ sleeps}](x)]$

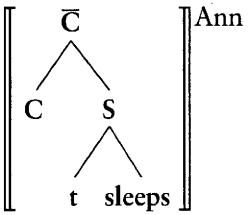
By the definition of λ-notation, this is equivalent to:



$= \lambda x \in D \,.\, x \text{ sleeps}.$

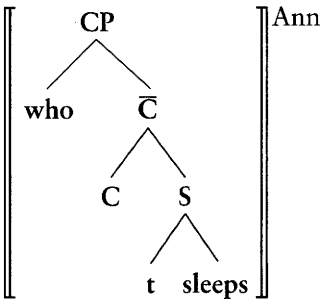
QED.

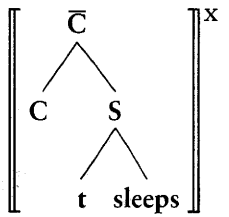
This is just as good a proof as the previous one. However, *it required a certain amount of foresight to write up*. We had to anticipate right at the beginning that we would need to work out the semantic values of the subtrees up to $\overline{C}$ *for an arbitrary assignment*. Had we just plunged in without thinking ahead, we would have been tempted to calculate something which subsequently turns out to be irrelevant: namely, the extension of $\overline{C}$ with respect to the *particular* assignment Ann. In that case, we would have wasted the first few steps of the proof to determine that



$= [\lambda y \in D \,.\, y \text{ sleeps}](\text{Ann}).$ [=: (i)]

This is perfectly correct, but it is not helpful for the continuation of the proof, because for the next node up, we derive:



$= \lambda x \in D \,.$ 

Now the fact established in (i) gives us only a tiny piece of partial information about which function it is that we have at the right-hand side of this equation:

(i) merely tells us that it is some function or other which *to Ann* assigns 1 iff she sleeps. It doesn't tell us how it behaves with arguments other than Ann. So that's not enough information to finish the proof. We are supposed to prove the equality of two *functions*, and for this it does not suffice to make sure that they assign the same value to Ann. Rather, we must show that they agree on *all* their arguments.

The moral of this is the following: If you already see clearly where you are headed, the bottom-up strategy can be as efficient and elegant as the top-down strategy. But if you are still groping in the dark or don't want to take any chances, top-down is the way to go. The advantage is that by the time you get to the lower nodes, you will already know which assignments it is relevant to consider.

Exercise

Some interesting issues arise when we bring together our new analysis of relative clauses and the previous chapter's Fregean treatment of definite descriptions. Suppose we embed a nondenoting definite in a relative clause:

(i) **John is a man who attended the 1993 Olympics.**

The intuitive status of (i) is the same as that of (ii).

(ii) **John attended the 1993 Olympics.**

Both are perceived as presupposition failures, since there were no Olympic Games in 1993. Our semantics from chapter 4 predicts that (ii) receives no truth-value. The same should come out for (i). Check whether it does by using the "pedantic" versions of the PA rule and the other composition principles, repeated here for convenience:

(12′) **Non-Branching Nodes (NN)**
If α is a non-branching node and β its daughter, then, for any assignment a, α is in the domain of $[\![\ ]\!]^a$ if β is. In this case, $[\![\alpha]\!]^a = [\![\beta]\!]^a$.

(13′) **Functional Application (FA)**
If α is a branching node and $\{\beta, \gamma\}$ the set of its daughters, then, for any assignment a, α is in the domain of $[\![\ ]\!]^a$ if both β and γ are, and $[\![\beta]\!]^a$ is a function whose domain contains $[\![\gamma]\!]^a$. In this case, $[\![\alpha]\!]^a = [\![\beta]\!]^a([\![\gamma]\!]^a)$.

(14′) **Predicate Modification (PM)**
If α is a branching node and $\{\beta, \gamma\}$ the set of its daughters, then, for any assignment a, α is in the domain of $[\![\ ]\!]^a$ if both β and γ are, and

$[\![\beta]\!]^a$ and $[\![\gamma]\!]^a$ are both of type <e,t>. In this case, $[\![\alpha]\!]^a = \lambda x : x \in D$ and x is in the domain of $[\![\beta]\!]^a$ and $[\![\gamma]\!]^a$. $[\![\beta]\!]^a(x) = [\![\gamma]\!]^a(x) = 1$.

(15′) **Predicate Abstraction (PA)**
If α is a branching node whose daughters are a relative pronoun and β, then $[\![\alpha]\!] = \lambda x : x \in D$ and β is in the domain of $[\![\ ]\!]^x$. $[\![\beta]\!]^x$.

Consider next a case where the definite in the relative clause *contains the variable*. An example is (iii).

(iii) **John is a man whose wife is famous.**

This involves so-called "pied-piping", the fronting of a larger constituent than the mere relative pronoun. Following many previous authors, we assume that pied-piping is essentially a surface phenomenon, and that the input structure for semantic interpretation is as if only the relative pronoun had moved:[13]

(iv) **John is a man who [t's wife is famous].**

We also assume that possessive constructions like "John's wife", "his wife", and in this case, "t's wife", are definites, headed by a non-overt equivalent of "the". Glossing over the details, we assume that the syntactic representation of "John's wife" (at the relevant level) is essentially "[the [wife (of) John]]". This can be straightforwardly interpreted by our semantic rules (assuming that "wife" has a meaning of type <e,<e,t>>). The prediction – adequate, it seems – is that "John's wife" denotes John's unique wife if he has a unique wife, and nothing otherwise. So a sentence like "John's wife is famous" is truth-value-less (a presupposition failure) if John is not married to any woman, or to several.

Back now to (iv). The analysis of "t's wife" is just like that of "John's wife", except, of course, that this time the denotation depends on the assignment. Assuming the "pedantic" versions of the composition principles, compute the denotations of the essential constituents of (iv). What happens if John is not married? What if nobody (in the salient universe of discourse) is married? At what point does the computation crash in each case?

5.3 Multiple variables

5.3.1 Adding "such that" relatives

Quine's discussion of relative clauses also includes a remark on the "such that" construction:

> The reason for permuting word order in forming relative clauses is to bring the relative pronoun out to the beginning or near it. The task can be exacting in complex cases, and is sometimes avoided by recourse to an alternative construction, the unlyrical "such that". This construction demands none of the tricks of word order demanded by "which", because it divides the two responsibilities of "which": the responsibility of standing in a singular-term position within the clause is delegated to "it", and the responsibility of signaling the beginning of the clause is discharged by "such that". Thus "which I bought" becomes "such that I bought it"; "for whom the bell tolls" becomes "such that the bell tolls for him".
>
> The "such that" construction is thus more flexible than the "which" construction.[14]

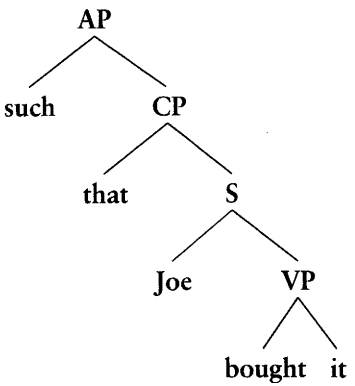
So the "such that" construction has a different syntax, but essentially the same semantics as ordinary relatives. It should be a routine matter, therefore, to extend our analysis to cover it. We just need to generalize our Traces Rule to pronouns, and to rewrite PA so that it treats a "such" like a relative pronoun. (The "that" is presumably the semantically vacuous complementizer again.)

(1) *Pronoun Rule* (new addition)
If α is a pronoun, then for any assignment $a \in D$ $(= D_e)$, $[\![\alpha]\!]^a = a$.

(2) *Predicate Abstraction* (revised)
If α is a branching node and β and γ its daughters, where β is a relative pronoun or β = "such", then $[\![\alpha]\!] = \lambda x \in D \,.\, [\![\gamma]\!]^x$.

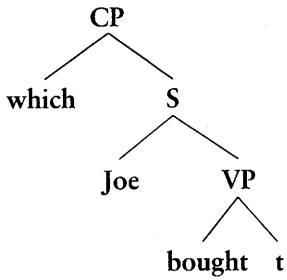
It is now easy to calculate a semantic value for a tree like (3),

(3)



and to prove that this "such that" phrase has exactly the same denotation as its *wh*-counterpart in (4).

(4)



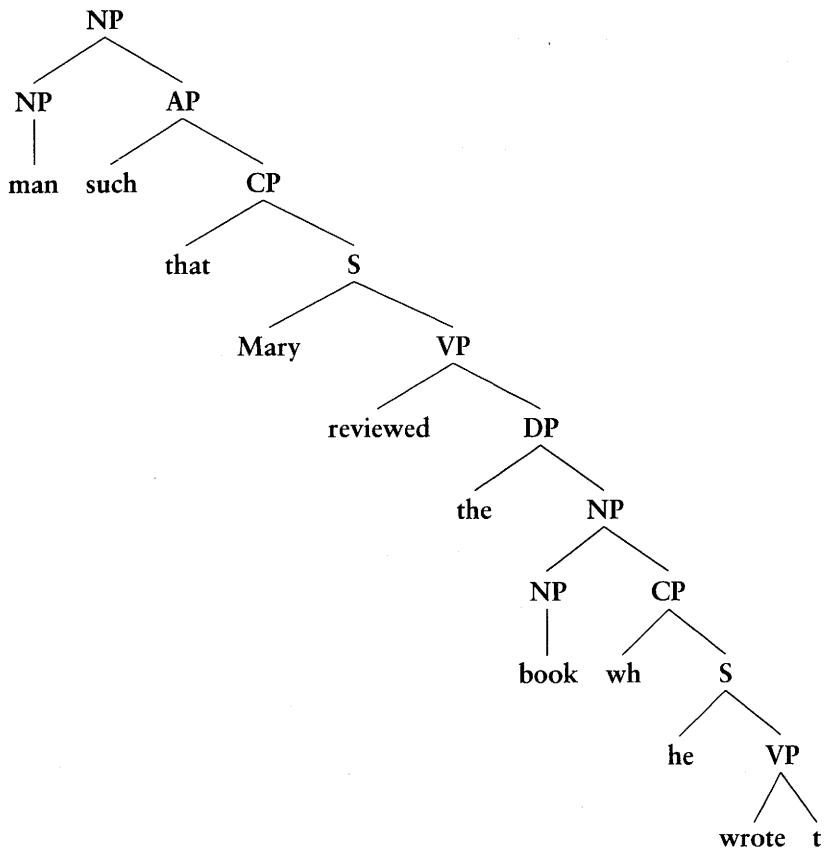
5.3.2 A problem with nested relatives

Our reason for introducing "such that" relatives is that it allows us to look at certain examples where one relative clause is nested inside another.[15] Let's see what interpretation we obtain for the following NP:

(5) **man such that Mary reviewed the book he wrote**

This phrase has the structure in (5′) and receives a meaning of type <e,t>.

(5′)



We encounter no problems of interpretability here, but the interpretation we derive under our current assumptions is very wrong.

Exercise

Prove that the following predictions are made by our current semantics:

(i) If there isn't a unique book that wrote itself, then (5) has no denotation under any assignment.
(ii) If Mary reviewed the unique book that wrote itself, then $[\![(5)]\!] = [\![\textbf{man}]\!]$.
(iii) If Mary didn't review the unique book that wrote itself, then $[\![(5)]\!] = \lambda x \in D \,.\, 0$.[16]

The meaning predicted, then, is not the meaning that (5) actually has in English. If we can express this strange predicted meaning in English at all, we have to use quite a different phrase, namely (6).

(6) **man such that Mary reviewed the book which wrote itself**

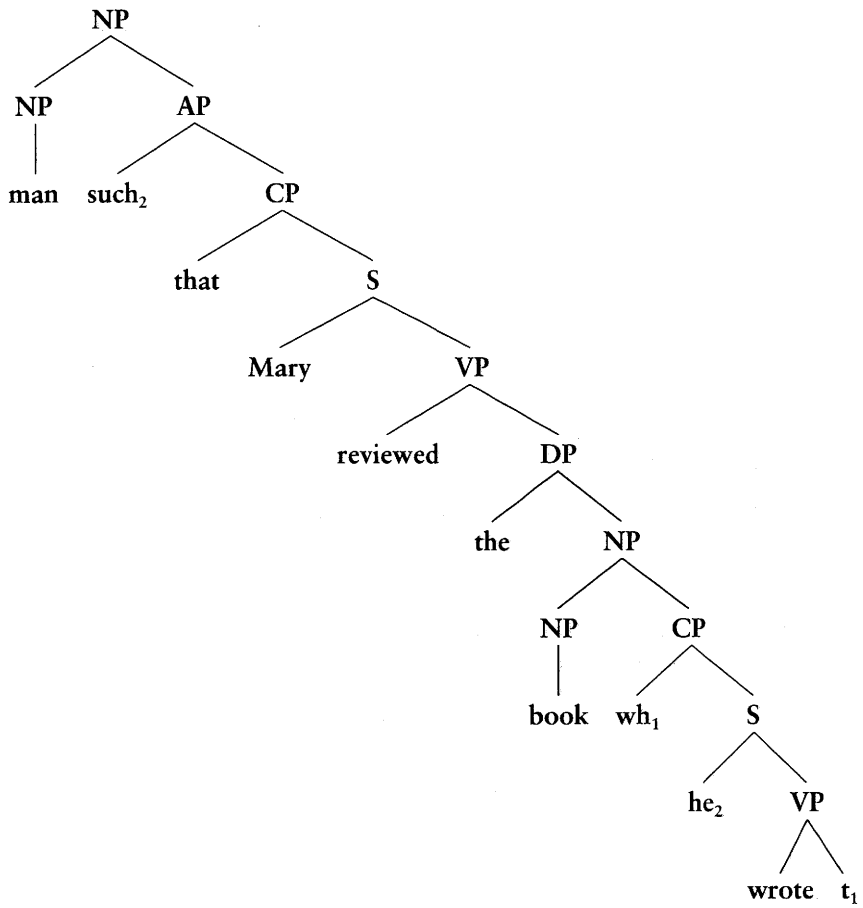
In order to rectify this inadequacy, we will have to reconsider the syntax as well as the semantics of variables.

5.3.3 Amending the syntax: co-indexing

The fact that our example (5) wrongly received the meaning of (6) points us to something that seems to be wrong with our present analysis: namely, that, informally speaking, the semantics pays no attention to which trace/pronoun is related to which *wh*-word/"such". Specifically, it apparently treated the "he" in (5) as if it was a trace of the "*wh*" right below "book". This suggests that part of the solution might be to employ syntactic representations which explicitly encode which trace or pronoun belongs with which relative pronoun or "such".

The standard notational device for this purpose is co-indexing by means of numerical subscripts. We will henceforth assume that the syntactic structure for (5) which constitutes the input to the semantic component is a bit richer than (5′) above. It looks rather like (5″) below.

(5″)



Of course, there are infinitely many other possible indexings for the same surface phrase. We will return to this point below. For the moment, our goal is to make sure that *if* (5) is represented as in (5″), then it receives the intuitively correct interpretation. Merely adding co-indexing in the syntax is not, of course, sufficient by itself to attain this goal. As long as our semantic rules fail to "see" the indices, co-indexing will do nothing to determine the appropriate denotation.

5.3.4 *Amending the semantics*

In order to write an appropriately index-sensitive set of semantic rules, we must first redefine what we mean by an "assignment". So far, an assignment was just an element of D (= D_e). This worked fine as long as we only considered examples in which each variable was contained in at most one relative clause. In our present, more complicated example (5), we have traces and pronouns that are

contained in both a smaller and a bigger relative clause. This situation requires a richer notion of assignment. Our new definition is (7):

(7) A *(variable) assignment* is a partial function from $\mathbb{N}$ (the set of natural numbers) into D.

For example, the following functions are assignments in this new sense:

(8) $\begin{bmatrix} 1 \rightarrow \text{John} \\ 2 \rightarrow \text{Mary} \end{bmatrix}$ $\begin{bmatrix} 1 \rightarrow \text{John} \\ 2 \rightarrow \text{John} \end{bmatrix}$ $[1 \rightarrow \text{John}]$

$\begin{bmatrix} 2 \rightarrow \text{John} \\ 5 \rightarrow \text{Mary} \\ 7 \rightarrow \text{Ann} \end{bmatrix}$

Given definition (7), $\varnothing$ (the empty set) comes out as an assignment too. $\varnothing$ is the (only) function with $\varnothing$ as its domain. Any function from $\varnothing$ into D, for example, would have to be a subset of $\varnothing \times D$, the Cartesian product of $\varnothing$ and D.[17] Since $\varnothing \times D = \varnothing$, the only subset of $\varnothing \times D$ is $\varnothing$, hence $\varnothing$ is the only function from $\varnothing$ into D. Since $\varnothing \subseteq \mathbb{N}$, $\varnothing$ qualifies as a partial function from $\mathbb{N}$ into D, hence as an assignment.

Since assignments are now functions, it makes sense to speak of their "domain". We write "dom(a)" to abbreviate "the domain of the assignment a". Assignments in the new sense assign potentially different individuals to different numerical indices. This allows us to replace our old rules for (unindexed) traces and pronouns by a new rule that is sensitive to the index.[18]

(9) *Traces and Pronouns Rule*
If α is a pronoun or a trace, a is a variable assignment, and $i \in \text{dom}(a)$, then $[\![\alpha_i]\!]^a = a(i)$.

(9) makes the following kinds of predictions:

(10) $[\![\mathbf{he}_2]\!]^{\begin{bmatrix} 1 \rightarrow \text{Sue} \\ 2 \rightarrow \text{Joe} \end{bmatrix}} = \begin{bmatrix} 1 \rightarrow \text{Sue} \\ 2 \rightarrow \text{Joe} \end{bmatrix}(2) = \text{Joe}$

$[\![\mathbf{t}_1]\!]^{\begin{bmatrix} 1 \rightarrow \text{Sue} \\ 2 \rightarrow \text{Joe} \end{bmatrix}} = \begin{bmatrix} 1 \rightarrow \text{Sue} \\ 2 \rightarrow \text{Joe} \end{bmatrix}(1) = \text{Sue}$

(9) also implies that a given trace or pronoun will not have a well-defined semantic value under just any assignment. For instance, $\mathbf{he}_2$ is not in the domain of $[\![\ ]\!]^{[1 \rightarrow \text{Joe}]}$, $[\![\ ]\!]^{[3 \rightarrow \text{Joe}]}$, or $[\![\ ]\!]^{\varnothing}$. As regards $[\![\ ]\!]^{\varnothing}$, *no* pronoun or trace is in its domain. Are any expressions at all? Yes, but only those which are in the domain of $[\![\ ]\!]^a$ for *every* a. This is the case, for example, for the lexical items. Recall our earlier convention, which we carry over into the new system:[19] When an expression

is in the domain of [[ ]], it is also in the domain of [[ ]]^a for all a. In fact, we will now think of "[[ ]]" as simply an abbreviation for "[[ ]]$^{\varnothing}$".

(11) For any tree α, $[[\alpha]] := [[\alpha]]^{\varnothing}$

To have a semantic value *simpliciter* means nothing more and nothing less than to have a semantic value under the empty assignment.

As before, assignment dependency is systematically passed up the tree when we construct phrases from lexical items and one or more traces and pronouns. The composition rules NN, FA, and PM can stay just as we formulated them before (though, of course, wherever they refer to "assignments", this now means something different than it used to). With their help, we can now calculate semantic values under given assignments for many larger phrases composed of lexical items and variables. For instance, we can prove the following:

(12) $\left[\left[\begin{array}{c} \text{he}_2 \\ \text{wrote} \quad t_1 \end{array}\right]\right]^{\left[\begin{smallmatrix} 1\rightarrow \text{“Barriers”} \\ 2\rightarrow \text{Joe} \end{smallmatrix}\right]}$

= 1 iff Joe wrote "Barriers"

Exercise

Prove (12).

The Predicate Abstraction rule will have to be revised. Before we can do this, we have to define one further technical concept, that of a so-called modified (variable) assignment:

(13) Let a be an assignment, $i \in \mathbb{N}$, and $x \in D$. Then $a^{x/i}$ (read: "a modified so as to assign x to i") is the unique assignment which fulfills the following conditions:
   (i) $dom(a^{x/i}) = dom(a) \cup \{i\}$,
   (ii) $a^{x/i}(i) = x$, and
   (iii) for every $j \in dom(a^{x/i})$ such that $j \neq i$: $a^{x/i}(j) = a(j)$.

(13) defines a possibly new assignment $a^{x/i}$ on the basis of a given assignment a. Clause (i) of (13) states that the domain of the new assignment contains the number i. If dom(a) contains i already, then, $dom(a) = dom(a^{x/i})$. Otherwise, the

index i has to be added. Clause (ii) states that the new assignment $a^{x/i}$ is a function that maps the number i to the individual x. If dom(a) contains i already, then $a^{x/i}$ might differ from a with respect to the individual assigned to i. Suppose a(5) = Mary, for example. Then $a^{Ann/5}(5)$ = Ann, hence $a(5) \neq a^{Ann/5}(5)$. Clause (iii) makes sure that the new assignment $a^{x/i}$ is indeed like the original assignment a, except for a possible difference regarding i.

Exercise

Determine $a^{Mary/2}$ for various concrete choices of a (for example, the assignments mentioned in (8) above).

Sample answers:

(14)
$$\begin{bmatrix} 1 \rightarrow \text{John} \\ 2 \rightarrow \text{Mary} \end{bmatrix}^{Mary/2} = \begin{bmatrix} 1 \rightarrow \text{John} \\ 2 \rightarrow \text{Mary} \end{bmatrix}$$

$$\begin{bmatrix} 1 \rightarrow \text{John} \\ 2 \rightarrow \text{John} \end{bmatrix}^{Mary/2} = \begin{bmatrix} 1 \rightarrow \text{John} \\ 2 \rightarrow \text{Mary} \end{bmatrix}$$

$$[1 \rightarrow \text{John}]^{Mary/2} = \begin{bmatrix} 1 \rightarrow \text{John} \\ 2 \rightarrow \text{Mary} \end{bmatrix}$$

As you can see from these examples:

The domain of $a^{Mary/2}$ always includes the number 2;
The domain of $a^{Mary/2}$ is either the same as the domain of a or larger by one element, depending upon whether a already happened to have 2 in its domain;
$a^{Mary/2}$ is either identical to a or differs from it in one argument, depending upon whether a already happened to map 2 to Mary.

Modified assignments can be modified further, so the notation can be iterated, and we get things like this:

(15) $$[[1 \rightarrow \text{John}]^{Mary/2}]^{Sue/1} = \begin{bmatrix} 1 \rightarrow \text{John} \\ 2 \rightarrow \text{Mary} \end{bmatrix}^{Sue/1} = \begin{bmatrix} 1 \rightarrow \text{Sue} \\ 2 \rightarrow \text{Mary} \end{bmatrix}$$

The bracketing here was added for clarity. In practice, we will write

$$[1 \rightarrow \text{John}]^{Mary/2,\ Sue/1}$$

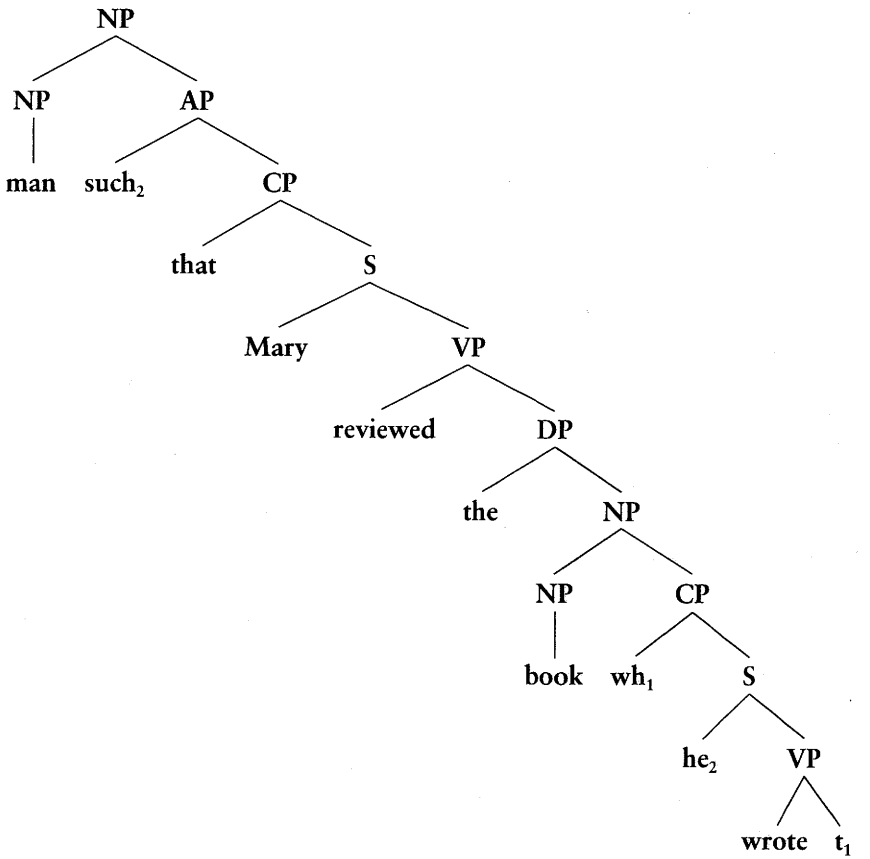
but it must be understood that this refers to the result of first modifying [1 → John] so as to assign Mary to 2, and then modifying *the result of that* so as to assign Sue to 1.

Reference to modified assignments allows us to manipulate the values of certain designated variables (that is, traces or pronouns) while leaving the values of all other variables intact. The usefulness of this device will be seen when we apply our new rule of Predicate Abstraction. This is given in (16).[20]

(16) *Predicate Abstraction Rule* (PA)
If α is a branching node whose daughters are β_i and γ, where β is a relative pronoun or "such", and $i \in \mathbb{N}$, then for any variable assignment a, $[\![\alpha]\!]^a = \lambda x \in D \,.\, [\![\gamma]\!]^{a^{x/i}}$.

Every application of the Predicate Abstraction Rule targets one variable (identified by its index) in an expression γ, and defines a function of type <e,t> by manipulating the value assigned to that particular variable. Let's see how this works by computing the denotation of our problematic "such that" clause in (5″) (repeated from above).

(5″)



What we want to show is that the constituent "such$_2$ that Mary reviewed the book wh$_1$ he$_2$ wrote t$_1$" denotes a certain function in $D_{<e,t>}$: namely, the one that maps to 1 exactly those individuals x such that Mary reviewed the book written by x. More precisely, we want to show that this constituent denotes this function independently of any assignment. So let's calculate:[21]

$[\![$**such$_2$ that Mary reviewed the book wh$_1$ he$_2$ wrote t$_1$**$]\!]$
= (by convention (11))
$[\![$**such$_2$ that Mary reviewed the book wh$_1$ he$_2$ wrote t$_1$**$]\!]^{\varnothing}$
= (by PA)
$\lambda x \in D$. $[\![$**that Mary reviewed the book wh$_1$ he$_2$ wrote t$_1$**$]\!]^{\varnothing^{x/2}}$
= (by definition (13) (assignment modification))
$\lambda x \in D$. $[\![$**that Mary reviewed the book wh$_1$ he$_2$ wrote t$_1$**$]\!]^{[2\rightarrow x]}$
= (by vacuity of **that** and three applications of FA)
$\lambda x \in D$. $[\![$**reviewed**$]\!]^{[2\rightarrow x]}([\![$**the**$]\!]^{[2\rightarrow x]}([\![$**book wh$_1$ he$_2$ wrote t$_1$**$]\!]^{[2\rightarrow x]}))([\![$**Mary**$]\!]^{[2\rightarrow x]})$
= (by convention (9) of section 5.2 and lexical entries for **review, the, Mary**)
$\lambda x \in D$. Mary reviewed the unique y such that
$[\![$**book wh$_1$ he$_2$ wrote t$_1$**$]\!]^{[2\rightarrow x]}(y) = 1$
= (by PM and lexical entry for **book**)
$\lambda x \in D$. Mary reviewed the unique y such that
y is a book and $[\![$**wh$_1$ he$_2$ wrote t$_1$**$]\!]^{[2\rightarrow x]}(y) = 1$

Exercise

Continue the computation started above.

5.4 What is variable binding?

In this section, we introduce definitions and theorems for some important notions related to variable binding. The whole section is necessarily more technical and more abstract than the previous parts of this book, and can be postponed[22] if you are satisfied with a merely informal understanding of the notion "variable binding" at this point.

5.4.1 Some semantic definitions

We have informally referred to traces and pronouns as "variables". What do we mean by this? The term "variable", though it clearly began its history as an (informal) semantic concept, is nowadays sometimes used in a purely syntactic sense, with explicit disclaimers that variables have any particular semantic interpretation in common. This practice may be unobjectionable in certain contexts, but not when matters of semantic interpretation and the syntax–semantics interface are at the very center of attention, as they are here in this book. We will therefore use the term "variable" (and various related terms) in a purely semantic sense. By this we don't mean that variables are not syntactic objects. They are. They are linguistic expressions. But what defines them as "variables" is not their shape or syntactic behavior, but the fact that they are interpreted in a certain way.

A variable in our sense is by definition something whose denotation varies with the assignment. More precisely:

(1) A terminal symbol α is a *variable* iff there are assignments a and a′ such that $[\![\alpha]\!]^a \neq [\![\alpha]\!]^{a'}$.

Consider in the light of this definition our Traces and Pronouns Rule from above:

(2) *Traces and Pronouns Rule*
If α is a pronoun or a trace, a is a variable assignment, and $i \in dom(a)$, then $[\![\alpha_i]\!]^a = a(i)$.

It follows directly from that rule (and plausible assumptions about D) that traces and pronouns are variables in the sense of (1). Terminal symbols that are not variables are called "constants":

(3) A terminal symbol α is a *constant* iff for any two assignments a and a′, $[\![\alpha]\!]^a = [\![\alpha]\!]^{a'}$.

Let's consider some further concepts that have to do with variables: in particular, the notions of "bound" versus "free" variables and of a "variable binder". These terms as well have been adapted to purely syntactic uses, but we are interested here in their semantic senses.

Roughly, what is meant by "variable binding" is any semantic operation which removes (or reduces) assignment dependency. By combining an expression whose denotation varies across assignments with one or more variable binders, we can create a larger expression whose denotation is assignment-invariant. This

is what happens when we build a relative clause. The embedded S-constituent (for example, "Joe likes t_1") has different semantic values under different assignments, but the complete relative clause (here "who$_1$ [Joe likes t_1]") has a fixed semantic value, which stays the same under all assignments. In our current semantics for English, Predicate Abstraction is in fact the only rule that accomplishes variable binding in this sense. Accordingly, the only items which qualify as variable binders are the ones which trigger the application of this rule: namely, indexed relative pronouns and "such". Here is a precise characterization of which expressions of a language L qualify as variable binders.[23]

(4) An expression α is a *variable binder* (in a language L) iff there are trees β (in L) and assignments a such that
   (i) β is not in the domain of $[\![\ ]\!]^a$, but
   (ii) some tree (of L) whose immediate constituents are α and β *is* in the domain of $[\![\ ]\!]^a$.

To appreciate how this definition works, it is useful to be aware of the following general fact:[24] Whenever any expression is in the domain $[\![\ ]\!]^a$ for any assignment a, it is also in the domain of $[\![\ ]\!]^{a'}$ for any larger assignment $a' \supseteq a$. So if an expression is not in the domain of $[\![\ ]\!]^a$ for a given a, this can only be because the domain of a is too small, never because it is too large. Hence the kind of assignment which satisfies conditions (i) and (ii) of (4) has to be an assignment whose domain is too small to interpret β, yet big enough to interpret the next higher node.

Consider an example: How does, say, "who$_2$" qualify as variable binder of English under (4)? Well, pick the expression "t_2 left" and the assignment $\varnothing$, for example. They meet the conditions (4)(i) and (4)(ii) on β and a respectively, since (i) "t_2 left" is not in the domain of $[\![\ ]\!]^{\varnothing}$, and (ii) the tree "[who$_2$ t_2 left]" *is* in the domain of $[\![\ ]\!]^{\varnothing}$. (We can show (i) and (ii) by applying the relevant lexical entries and composition rules.) For a negative example, why is, say, "the" *not* a variable binder of English? Because we can prove from its lexical entry and the FA rule that whenever an NP β is not in the domain of $[\![\ ]\!]^a$, then a DP consisting of the determiner "the" and the NP β isn't either.

The related notions of "bound" and "free" occurrences of variables can be defined along similar lines. These terms (unlike the ones defined so far) apply to particular *occurrences* of expressions in a tree, not to the expressions (considered as types) themselves. We will see that the same variable may have bound and free occurrences in a given tree. The basic intuition behind the "free" versus "bound" distinction is this: A variable is "free" in a tree if that tree can only be interpreted under a specified assignment to this variable. For instance, the object trace $\mathbf{t_1}$ is *free* in the trees [$_{VP}$ **likes** $\mathbf{t_1}$], [$_{S}$ **Mary likes** $\mathbf{t_1}$], [$_{S}$ **it is not the case that he$_2$ likes** $\mathbf{t_1}$], and [$_{NP}$ **man who$_2$** $\mathbf{t_2}$ **likes** $\mathbf{t_1}$], since all these trees have well-defined

denotations only under assignments whose domains include 1 and thus specify a value for "t_1". By contrast, the object trace "t_1" is *bound* (= *not free*) in the trees [$_{CP}$ **who$_1$ Mary likes t$_1$**], and [$_{NP}$ **man who$_1$ he$_2$ likes t$_1$**], since these trees do have denotations under assignments that exclude 1. An example with bound and free occurrences of the same variable in different places in the tree would be [$_S$ **t$_1$ likes the man who$_1$ t$_1$ left**], in which the first occurrence of "t_1" is free and the second bound.

To formulate a precise definition, we need a way of referring to occurrences of expressions. Let's assume that the occurrences of each given expression α in a tree can be numbered α^1, α^2, etc. (say, from left to right).[25]

(5) Let α^n be an occurrence of a variable α in a tree β.
   (a) Then α^n is *free* in β if no subtree[26] γ of β meets the following two conditions:
      (i) γ contains α^n, and
      (ii) there are assignments a such that α is not in the domain of $[\![\ ]\!]^a$, but γ is.
   (b) α^n is *bound* in β iff α^n is not free in β.

Exercise

Apply these definitions to the examples given in the text right above.

It is a consequence of definition (5) that α^n can be bound in β only if β contains a variable binder in the sense of definition (4). In fact, something more specific follows: namely, that β must contain an occurrence of a variable binder in β which *c-commands*[27] α^n. Let us examine why this is so.

Suppose α^n is bound in β. Then definition (5) implies that there is some subtree of β which fulfills the conditions (i) and (ii). Let γ be the *smallest* such subtree of β. Let δ be that daughter of γ which contains α^n, and ε that daughter of γ which doesn't contain α^n. Let a be an assignment such that γ, but not α, is in the domain of $[\![\ ]\!]^a$. By assumption, γ is the smallest subtree of β which meets (i) and (ii), so δ (which is smaller) doesn't. In particular, δ must fail (ii), since it does meet (i) by assumption. Hence it cannot be the case that δ is in the domain of $[\![\ ]\!]^a$. But this implies that ε meets the definition of "variable binder" in (3). QED.

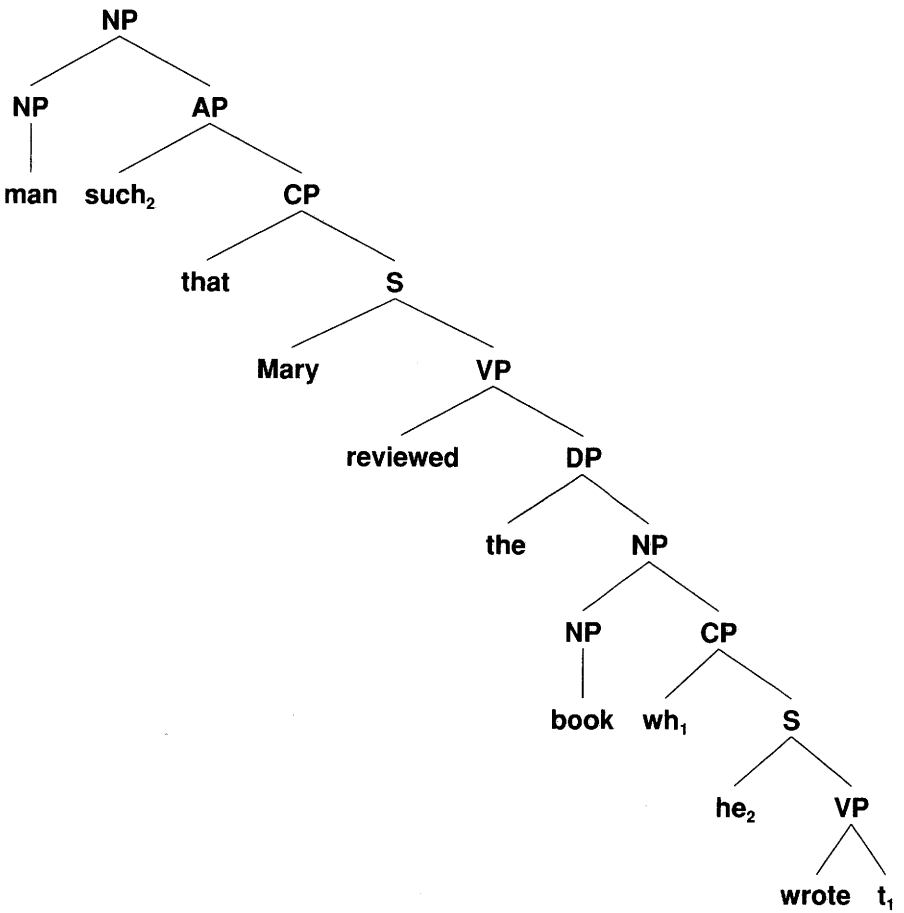
We have shown that for every bound variable there is a variable binder which is responsible for its being bound. This being so, it makes sense to define a 2-place relation "binds", which obtains between an occurrence of a variable binder and an occurrence of a variable:

(6) Let β^n be a variable binder occurrence in a tree γ, and let α^m be a variable occurrence in the same tree γ which is bound in γ.
Then β^n *binds* α^m iff the sister of β^n is the largest subtree of γ in which α^m is free.

It follows directly from (4), (5), and (6) that (i) every bound variable is bound by a unique binder, and that (ii) no free variable is bound by anything. (By (ii), we mean more precisely: if a variable is free in γ, then nothing in γ binds it.)

Exercise

What binds what in the following structure:



Show how the present definition applies to each binder–bindee pair.

5.4.2 Some theorems

The concepts we have introduced in this section provide a convenient and widely used terminology in which we can state a number of generalizations which turn out to be predicted by our current semantics for English phrase structures. For example, we can give a precise characterization of the syntactic configurations in which variables get bound. First, we can prove that there is no binding without co-indexing:

(7) If β binds α, then β and α are co-indexed.

Exercise

Prove (7).

Using (7) and our earlier definition of "variable binding", we can state necessary and sufficient conditions on the syntactic configurations in which variables get bound:

(8) β binds α if and only if
- (i) α is an occurrence of a variable,
- (ii) β is an occurrence of a variable binder,
- (iii) β c-commands α,
- (iv) β is co-indexed with α, and
- (v) β does not c-command any other variable binder occurrence which also c-commands and is co-indexed with α.

Exercise

Prove (8).

We can also state a precise theorem about how the extent to which the denotation of a tree depends on the choice of assignment correlates with the presence of free variables in that tree. The general version of this theorem is (9), and a special case is (10).

(9) If a tree γ contains no free occurrences of variables indexed i, then for all assignments a and all $x \in D$:
either γ is in the domain of neither $[\![\ ]\!]^a$ nor $[\![\ ]\!]^{a^{x/i}}$,
or it is in the domain of both, and $[\![\gamma]\!]^a = [\![\gamma]\!]^{a^{x/i}}$.

(10) If a tree γ contains no free variables, then it either has no semantic value under any assignment, or else it has the same one under all of them.

Here is a proof-sketch for (9). We begin by observing that (9) holds trivially when γ is a *terminal* node. Then we show that, if γ is a *non-terminal* node, and if we can take for granted that all of γ's daughters conform to (9), it follows that γ itself will too. To do this, we consider five cases. The first case (trivial) is that none of our composition principles is applicable to γ. The second case (also trivial) is that NN applies to γ. The remaining three cases correspond to the composition principles FA, PM, and PA. We first show that if FA applies to γ, then γ must conform to (9) whenever both its daughters do. Then we do the same for PM, and finally for PA. When we are done with all these subproofs, we have shown that, however γ is constructed, it must conform to (9).

Why is it useful to be aware of laws like (7)–(9)? It can often save us lengthy calculations when we want to assess whether a given syntactic representation does or does not represent a given denotation. For instance, by relying on (7) and (9) we can tell very quickly that the predicate **such$_1$ that Mary reviewed the book wh$_2$ he$_2$ wrote t$_2$** denotes a constant function (and thus cannot possibly single out a non-empty proper subset of the men). Using (8), we determine that the S beginning with **Mary** contains no free variable, and given (9), this implies that this S is either true under all assignments or false under all assignments. We need not derive predictions like this by doing all our semantic calculations from scratch with each new example. Once the appropriate general theorems have been proved, many questions about the interpretation of a given syntactic structure can be answered simply by taking a glance at it. Experienced semanticists use such shortcuts all the time, and we will increasingly do so in the later chapters of this book. Still, we must never lose sight of the fact that all predictions about which structure means what must be deducible from the lexical entries and composition principles, and from those alone. If they are not, they are not predictions at all.

5.4.3 Methodological remarks

We have stressed that we are interested in the *semantic* senses of the concepts "variable", "bound", "free", etcetera. We acknowledged in passing that other

authors use these terms in purely syntactic senses, but have made it sound as if this was a relatively recent development in the syntactic literature, and that tradition was on our side. "What tradition?," you may be wondering, especially if you have some background in formal logic. Logic books invariably give purely syntactic definitions of these terms. For instance, a typical presentation of the syntax of Predicate Logic (PL) may begin with the definition of a vocabulary, including a clause like: "the variables of PL are x, y, z, x_1, y_1, z_1, x_2, y_2, z_2, . . .". So the term "variable" is defined simply by a list, or by a specification of the shapes of variables. Typical definitions of "bound" and "free" for variables in PL formulas read like this: "An occurrence of a variable α in a formula ϕ is *free in* ϕ iff this occurrence of α does not fall within any subformula of ϕ that has the form $\forall\alpha\psi$ or $\exists\alpha\psi$." If this is not a purely syntactic definition, then what is? Apart from superficial differences in terminology (for example, logicians don't say "c-command"), the standard logic-book definition of "variable binding" amounts to exactly the biconditional that we presented as a "theorem" in (8) above – except that the term "variable binder" is typically replaced by a list of the relevant symbols: namely, "$\forall\alpha$" and "$\exists\alpha$" in the case of PL.

So it looks as if we don't really have tradition on our side when we insist that "variable" and "binding" be defined in terms of their semantic import, or when we say that (8) describes the empirical predictions of a particular semantic theory and is not true by mere definition.

However, the picture changes somewhat if we attend not just to the official definitions, but also to the informal terminological practices in the logic literature. While the explicit definitions for "variable," "bound", etcetera are syntactic definitions, there is also an informal, but very firmly established, consensus among logicians that it would be inappropriate to employ these terms for expressions that do not share essentially the same semantics. This emerges more clearly if you don't just look at presentations of standard systems like PL, but observe the terminological choices that a logician will make when contemplating innovations or comparing different systems. No logician will call a "variable" something that is interpreted as a constant (in the sense captured by a semantic definition like our (3)). And no logician would alter the definition of "bound"/"free" (perhaps with the aim of making it simpler) without proposing exactly concomitant changes in the semantics. (The definition is clearly not simple or natural from a syntactic point of view; this in itself is an indication of the fact that the concept it aims to formalize is not a syntactic, but a semantic, one.) We are not claiming that all commonly accepted uses of the relevant terminology are compatible with the exact definitions we have formulated here. In fact, perhaps none of them are, since these particular definitions make sense only in the context of a particular formalization of the semantics of variables. Even relatively superficial technical changes would require revisions in the definitions. Still, we think that the concepts we have defined can be recognized as formal

counterparts of the informal concepts which have guided the terminological practice (if not the explicit definitions) of logicians and formal semanticists.

So, as far as the question "What tradition?" is concerned, we note that there is more to a logician's concept of variable-hood and binding than what is revealed in the letter of the standard definitions. We have tried to elucidate some of this. To what end? Why do we, as linguists, feel the need to keep syntactic and semantic terminology neatly apart where logicians have been quite content to conflate them? This has to do with a fundamental difference between formal and natural languages, or between logic and linguistic semantics. The formal languages defined by logicians are specifically *designed* to receive a certain semantic interpretation. Their vocabulary and syntax are deliberately set up in such a way that vocabulary items that are interpreted alike are also alike in shape and distribution. It would be pointless for the inventor of a logical language to introduce syntactically distinct vocabulary items (say, "pronouns" and "traces", or "nouns" and "verbs") only to give them identical interpretations. Therefore, the syntactic categories of these artificial languages correspond straightforwardly to classes of expressions with shared semantic properties. It is not a discovery or a matter of possible controversy that this should be so. It is therefore harmless, and even convenient, to use the very terms that informally characterize a certain semantic behavior as labels for the artificial syntactic categories. But what's harmless and convenient for logicians may be confusing and counterproductive for linguists. When it comes to theorizing about natural language, we don't know ahead of the endeavor what general properties we will find. Perhaps natural languages will turn out to have some of the design features of logical languages, such as a more or less simple correspondence (or even a one-to-one match) between syntactic and semantic categories. This could be an interesting finding, and it looks like an interesting working hypothesis to explore. But it is hardly obvious, and it is definitely controversial. Whether we want to argue for it or against it, we are better off with a terminology that allows us to state the issue than with one that equivocates.

5.5 Interpretability and syntactic constraints on indexing

In this section, we will reflect on the proper assignment of indices to pronouns and traces. Look at the analysis of our example from the previous section, for example.

(1) **man such that Mary reviewed the book wh he wrote t**

We succeeded in predicting that this NP *can* have the reading it intuitively has. That is, we found a structure for (1) which, by our revised semantic rules, demonstrably expresses the intended meaning:

(2) **man such$_1$ that Mary reviewed the book wh$_2$ he$_1$ wrote t$_2$**

But we have yet to show that our grammar correctly predicts the meaning of (2) to be the *only* reading of (1). In other words, we have yet to make sure that we don't generate any additional, intuitively unavailable, meanings for (1).

The *prima facie* problem here is that the indices, which we saw play a crucial role in determining denotation, are not overt on the surface. So in principle, the surface structure of (1) could be associated with as many different structures as there are different possible indexings. Many of the infinitely many options that arise here are semantically equivalent. Still, there are quite a few non-equivalent choices for assigning indices to the four items "such", "wh", "he", and "t" in (1). What about (3), for example?

(3) **man such$_1$ that Mary reviewed the book wh$_2$ he$_2$ wrote t$_1$**

(3) has a meaning that is easy to express in English: namely, the meaning of "man such that Mary reviewed the book *that wrote him*". Why can (1) not be read in this way? Because, we assume, the structure in (3) is syntactically ill-formed. It is simply not made available by the syntactic component of the grammar. If it were well-formed, we claim, then (1) could be understood in the sense indicated. This is what our semantics commits us to, at any rate, and as far as we can see, it is consistent with the evidence.

From a syntactician's point of view, the assumption that (3) is syntactically ill-formed is unobjectionable. In fact, it exhibits simultaneous violations of several commonly proposed syntactic constraints. One problem is that "such" in (3) binds a trace instead of a pronoun. That this in itself leads to ungrammaticality can be seen in simpler examples like (4).

(4) **the man such$_1$ that he$_1$/*t$_1$ left**

Another problem is that the relative pronoun in (3) binds a pronoun rather than a trace. This can also be seen to be a sufficient reason for ungrammaticality, at least when the pronoun is so close to its binder:

(5) **the man who$_1$ *he$_1$/t$_1$ left**

Over longer distances, English sometimes tolerates such "resumptive pronouns" fairly well (for example, ? "the man who$_1$ Mary wonders where he$_1$ went"), and other languages allow them freely, even in local configurations like (5). This sort

of cross-linguistic and language-internal variation makes it all the more plausible that our semantics is on the right track in giving pronouns and traces the same interpretation. It is not inconceivable a priori that the different distribution of pronouns and traces in English ultimately derives from some nonobvious interpretive difference between the two, or that there could be differences between the vocabularies of individual languages in this regard. But at the current state of linguistic research, there is less evidence for this than for the hypothesis that the distributional regularities have syntactic explanations compatible with an indiscriminate semantics.

Yet another thing that seems to be wrong with (3) has to do with the distribution of gender features: the masculine feature of "he" somehow clashes with the neuter feature of "book". Again, simpler examples than (3) illustrate the phenomenon more clearly:

(6) **the book such$_1$ that Mary reviewed *him$_1$/it$_1$**

It is less clear than in the cases of (4) and (5) that what we are looking at here is a matter of *syntactic* well-formedness. We *could* treat it as that, by laying down an appropriate set of syntactic agreement principles. But lay opinion points towards a different approach. The naive answer to what is wrong with "the book such that Mary reviewed him" is that "him" can't refer to an inanimate thing like a book, which sounds like a semantic explanation. Could this be spelled out in our theory? Taken literally, we cannot accept the naive story. Bound variables don't *refer* to individuals in the first place. As we have argued at the beginning of the chapter, the "it" in the good variant of (6) does not refer to a book either. But there seems to be a way of rephrasing the basic intuition in the light of this objection: suppose "him$_1$" cannot denote an inanimate (or non-male) individual *under any assignment*. More precisely, for any assignment a such that a(1) is not male, "him$_1$" is not in the domain of $[\![\ ]\!]^a$. What would this predict for (6)?

The immediate prediction is that "Mary reviewed him$_1$" will have a truth-value only under those assignments which map 1 to a male. For assignments a which don't meet this condition, $[\![$**Mary reviewed him$_1$**$]\!]^a$ is undefined. "Mary reviewed it$_1$", by contrast, gets a truth-value only under those assignments which map 1 to a nonhuman. What happens, then, when we apply the predicate abstraction rule to compute the denotation of $[\![$**such$_1$ that Mary reviewed *him$_1$/it$_1$**$]\!]$? Here we need the pedantic version, which produces partial functions:

(7) Predicate Abstraction (pedantic version)
If α is a branching node whose daughters are β_i and γ, where β is a relative pronoun or "such", and $i \in \mathbb{N}$, then for any variable assignment a, $[\![\alpha]\!]^a = \lambda x : x \in D$ and γ is in the domain of $[\![\ ]\!]^{a^{x/i}}$. $[\![\gamma]\!]^{a^{x/i}}$.

Given (7), we obtain:

(8) (a) $[\![\textbf{such}_1 \textbf{ that Mary reviewed him}_1]\!]$ =
$\lambda x : x \in D$ and x is male. Mary reviewed x.
(b) $[\![\textbf{such}_1 \textbf{ that Mary reviewed it}_1]\!]$ =
$\lambda x : x \in D$ and x is nonhuman. Mary reviewed x.

If we combine $[\![\textbf{book}]\!]$ with the partial function in (8a), we need the pedantic version of the Predicate Modification rule:

(9) Predicate Modification (pedantic version)
If α is a branching node and $\{\beta, \gamma\}$ the set of its daughters, then, for any assignment a, α is in the domain of $[\![\ ]\!]^a$ if both β and γ are, and $[\![\beta]\!]^a$ and $[\![\gamma]\!]^a$ are both of type <e,t>. In this case, $[\![\alpha]\!]^a = \lambda x : x \in D$ and x is in the domain of $[\![\beta]\!]^a$ and $[\![\gamma]\!]^a$. $[\![\beta]\!]^a(x) = [\![\gamma]\!]^a(x) = 1$.

Given (9), the denotation of **book such$_1$ that Mary reviewed him$_1$** is a function that does not map anything to 1, regardless of what the facts are: only male individuals are in its domain, and of those, none are books. Therefore, the definite DP **the book such$_1$ that Mary reviewed him$_1$** can never get a denotation. Combining $[\![\textbf{book}]\!]$ with the function in (8b), on the other hand, yields a function which maps all books that Mary reviewed to 1, and all nonhuman things which are either not books or are not reviewed by Mary to 0 (and is undefined for humans). Under the appropriate circumstances (that is, if there happens to be a unique book reviewed by Mary), **the book such$_1$ that Mary reviewed it$_1$** thus denotes.

This is only a sketch of a semantic account of gender agreement, but it looks as if it could be made to work. For the purposes of this book, we may leave it open whether feature mismatches as in (3) and (6) are ruled out by the syntax or by the semantics.

Besides (2) and (3), there are many other conceivable indexings of (1). But (equivalent variants aside), these two were the only possibilities in which each of the two binders binds a variable. What about indexings which don't have this property, say (10)?

(10) **man such$_1$ that Mary reviewed the book wh$_1$ he$_1$ wrote t$_1$**

Here the **such$_1$** binds no variable. (This is so despite the fact that it c-commands and is co-indexed with two variables. Those variables are already bound by the lower **wh$_1$**.) It is a so-called vacuous binder. Syntacticians have proposed that vacuous binding configurations are generally ill-formed:

(11) Each variable binder must bind at least one variable.

Again, simpler examples than (10) are more appropriate to motivate (11), since (10) violates other conditions as well (see below). Consider (12).

(12) (a) ***the man such that Mary is famous**
(b) ***the man who Mary is famous**

Here there simply are no variables present at all, so the binders in (12a) and (12b) will be vacuous under any possible indexing. Plausibly, this is the reason why these examples are ungrammatical.

Notice that (12a) and (12b) are not uninterpretable. Our semantics predicts, in fact, that if Mary happens to be famous and there happens to be just one man in D, then (12a) and (12b) denote this unique man. Otherwise they denote nothing. So we predict that (12a) and (12b) are tantamount to the simple DP **the man**, except that they convey an additional presupposition: namely that Mary is famous. Similarly, we do derive a well-defined meaning for (10) – in fact, the same meaning that we derived for (1) before we introduced the refinements of section 5.3 – see above. It is not *prima facie* inconceivable that natural language might have expressed such meanings, and might have expressed them in these forms. But (12a) and (12b) cannot be used with this (or any) meaning. So we need a syntactic constraint like (11) to exclude them.

(11) takes care of ruling out (10) and all other indexings of (1) in which one or both binders wind up vacuous. As we already noted, we have thus considered all possible indexings of (1), since there are only as many variables as binders in this example. Other examples, however, offer additional options, and allow us to illustrate further syntactic constraints on indexing.

For instance, neither of the two indexings we give in (13) below violates any of the syntactic constraints we have mentioned so far. Yet only one represents a meaning that is available for the corresponding surface string.

(13) **the man who$_1$ t$_1$ talked to the boy who$_2$ t$_2$ visited him$_{1/*2}$**

On the starred indexing, our semantics interprets (13) as denoting the man who talked to the boy who visited *himself*. This is a typical violation of a well-known binding constraint that says that a non-reflexive pronoun cannot be co-indexed with a c-commanding position in the same minimal clause. The precise formulation of the relevant constraint has been a topic of much syntactic research. Our aim here is not to contribute to this, but just to point out where binding constraints fit into our overall picture: The structures that violate such binding constraints are no less interpretable than those that conform to it. Binding constraints belong to syntax: they talk about purely formal aspects of syntactic representations, specifically about indices, c-command, and morphological properties of indexed DPs. Any semantic predictions they imply – that is,

predictions about possible and impossible meanings for a given sentence – come about indirectly, and depend crucially on the semantics that interprets indexed structures.

In sum, we argued that our semantics for variables and variable binders makes correct predictions about the meanings of a wide range of relative clause constructions, provided it is combined with suitable syntactic constraints which cut down on the number of possible indexings. The constraints which we found it necessary to appeal to all appeared to be well-established in syntactic theory.

Notes

1 Quoted from W. V. O. Quine, *Word and Object* (Cambridge, Mass., MIT Press, 1960), pp. 110–11; emphasis and footnotes added.

2 Quine distinguishes between *absolute general terms* and *relative general terms*. By an "absolute (general) term" he means a 1-place predicate, i.e., an expression that is true or false of an object. In our terms, this would be an expression with a meaning of type <e,t>, e.g., an intransitive verb or common noun. By a "relative (general) term" he means a 2- (or more) place predicate. His examples are "part of", "bigger than", "exceeds".

3 A *singular term* is any expression with a denotation in D_e (e.g., a proper name).

4 An "adjective" in Quine's sense seems to be anything that can modify a noun. It needn't have a head of the syntactic category A.

5 *Conversion* is the operation that switches the two arguments of a 2-place predicate; e.g., it maps "taller" to "shorter", and "buy" to "be-bought-by". *Application* is the combination of a 2-place predicate with an argument to yield a 1-place predicate; e.g., it forms "likes John" from "likes" and "John".

6 These judgments about (1) and (2) must be understood with the usual qualification: viz., that we pretend that the domain D contains only those individuals which are considered relevant in the context in which these sentences are uttered. (2) does not really presuppose that there is exactly one house *in the world*. But it does presuppose that only one house is under consideration, and this presupposition makes it appreciably odd in some utterance contexts where (1) would be entirely natural.

7 We will not always write both the *wh*-word and "that" in the trees below.

8 Richard Montague treated pronouns, proper names, and quantifier phrases as "term phrases". Our current NPs would correspond to Montague's "common noun phrases". See R. Montague, "On The Proper Treatment of Quantification in Ordinary English," in R. H. Thomason (ed.), *Formal Philosophy. Selected Papers by Richard Montague* (New Haven, Yale University Press, 1974), pp. 247–70. The term "DP" is due to S. Abney, "The English Noun Phrase in its Sentential Aspect" (Ph.D. dissertation, MIT, 1987).

9 Remember that we are here concerned with restrictive relatives. The criticism that follows would not apply to nonrestrictives.

10 See section 4.5, where we discussed the analogous bracketing with a PP-modifier.

11 The fully explicit versions of (12)–(14), which take possible undefinedness into account, are as follows:

(12′) If α is a non-branching node and β its daughter, then, for any assignment a, α is in the domain of $[\![\ ]\!]^a$ if β is. In this case, $[\![\alpha]\!]^a = [\![\beta]\!]^a$.

(13′) If α is a branching node and $\{\beta, \gamma\}$ the set of its daughters, then, for any assignment a, α is in the domain of $[\![\ ]\!]^a$ if both β and γ are, and $[\![\beta]\!]^a$ is a function whose domain contains $[\![\gamma]\!]^a$. In this case, $[\![\alpha]\!]^a = [\![\beta]\!]^a([\![\gamma]\!]^a)$.

(14′) If α is a branching node and $\{\beta, \gamma\}$ the set of its daughters, then, for any assignment a, α is in the domain of $[\![\ ]\!]^a$ if both β and γ are, and $[\![\beta]\!]^a$ and $[\![\gamma]\!]^a$ are both of type <e,t>. In this case, $[\![\alpha]\!]^a = \lambda x : x \in D$ and x is in the domain of $[\![\beta]\!]^a$ and $[\![\gamma]\!]^a$. $[\![\beta]\!]^a(x) = [\![\gamma]\!]^a(x) = 1$.

12 Pedantic version:

(15′) If α is a branching node whose daughters are a relative pronoun and β, then $[\![\alpha]\!] = \lambda x : x \in D$ and β is in the domain of $[\![\ ]\!]^x$. $[\![\beta]\!]^x$.

13 There are interesting issues about the syntax and semantics of movement here which we don't want to take up at this point. The issue we are pursuing in this exercise does not really depend on one's analysis of pied-piping. This just happens to be the most straightforward kind of example that we can treat at this stage.

14 Quine, *Word and Object*, p. 112.

15 Trying to paraphrase the "such that" relative by an ordinary *wh*-relative in such cases typically leads to ungrammaticality, at least in English: e.g., (5) would come out as *"man who Mary reviewed the book wrote". This can be improved a bit by inserting a resumptive pronoun for the *wh*-trace: ?"man who Mary reviewed the book he wrote". We will return to examples of this sort and the general issues which they raise about the division of labor between syntax and semantics at the end of this chapter.

16 λx . 0 is the function that maps every individual to the truth-value 0. In other words, it is the characteristic function of the empty set.

17 For any sets A and B, the Cartesian product of A and B, written as "A × B", is defined as $\{<x, y> : x \in A$ and $y \in B\}$.

18 Pedantic version:

(8′) If α is a pronoun or a trace, a is a variable assignment, and $i \in$ dom(a), then α_t is in the domain of $[\![\ ]\!]^a$, and $[\![\alpha_i]\!]^a = a(i)$.

19 Pedantic version: α is in the domain of $[\![\ ]\!]$ only if α has a semantic value under all assignments, and the same one under all assignments.

20 Pedantic version: If α is a branching node whose daughters are β_i and γ, where β is a relative pronoun or "such", and $i \in \mathbb{N}$, then for any variable assignment a, $[\![\alpha]\!]^a = \lambda x : x \in D$ and γ is in the domain of $[\![\ ]\!]^{a^{x/i}}$. $[\![\gamma]\!]^{a^{x/i}}$.

21 In the presentation of this calculation, we are writing mere strings between the double brackets. Strictly speaking, what should appear there, of course, are the corresponding *phrase structure trees*. This is merely a shorthand. We mean each string to stand for the subtree that dominates it in (5″) above. So please look at that diagram (5″) in order to follow the calculation.

22 The semantic notion of "variable binding" will be needed in chapter 10.

23 (4) presupposes that all trees are at most binary-branching. This is just to simplify matters. (4) is a special case of a more general definition, according to which α is a variable binder iff there are trees $\beta_1, \ldots, \beta_n$ (for some $n \geq 1$) and assignments a such that (i) none of the β_i are in the domain of $[\![\ ]\!]^a$, but (ii) a tree whose immediate constituents are α and $\beta_1, \ldots, \beta_n$ *is* in the domain of $[\![\ ]\!]^a$.

24 We do not give a proof of this law here, but take a minute to think about why it holds true. For lexical items, it follows from convention (9) in section 5.2. For traces and pronouns, it follows directly from the Traces and Pronouns Rule. This takes care of all possible terminal nodes. Now if we take any one of our composition rules, we can show that, as long as the law is true of all the daughter nodes, the application of the rule will ensure that it is true of the mother node as well.

25 We use superscripts here to avoid any confusion with the subscript indices, which are parts of the variables (*qua* expressions) themselves.

26 "Subtree" is understood in such a way that β counts as a subtree of itself.

27 "C-command" in the sense of the definition: x c-commands y iff x is sister to a node which dominates y.