Pronouns and variable binding

1 Pronouns

Recall the "free" vs "bound" terminology for variables. This is useful for natural language sentences as well:

(1) a. John likes Mary. — a sentence with no variables; not context-sensitive

b. John likes him. — a sentence with a variable; context-sensitive

c. Every boy likes himself. — a sentence with a variable; *not* context-sensitive

We'll formalize this by giving each pronoun a numerical *index*. Let's call the mapping between free variables and their values *assignment*. We'll compute denotations relative to an *assignment function*, which is a function from the set of indices ($\mathbb{N}$) to D_e.

(2) **Pronouns Rule (to be replaced later):**
If α is a pronoun, g is a variable assignment, and $g(i)$ is defined, then $[\![\alpha_i]\!]^g = g(i)$.

(3) Suppose g is a function and $g(3) = \text{Sam} \in D_e$.

a. $[\![\text{him}_3]\!]^g = g(3) = \text{Sam}$

b. $[\![\text{John likes him}_3]\!]^g = \text{Like(John, } g(3)) = \text{Like(John, Sam)}$

Q: Does it matter what g returns for other values in (3)?
A: No. It might even be undefined for other values.

Q: Why did we use 3? Does the number matter?
A: The choice of number was arbitrary, but it is important whether or not we reuse numbers:

(4) a. He_2 thinks that he_2 is smart.

b. He_2 thinks that he_7 is smart.

Q: Does the assignment function affect other parts of the sentence?
A: No. "John" and "likes" are *constants*, meaning their values are the same no matter the assignment: for any assignment function f, $[\![\text{John}]\!]^f = \text{John}$.

Warning: There's a section of H&K (pp. 92–109) where they just use notation like $[\![\text{him}]\!]^{\text{John}} = \text{John}$, which only accommodates one variable at a time, but then they introduce their actual notation on page 110, which we use here.

2 Rules with assignments

In order to work with assignment functions, we need to modify all our existing rules so that they pass assignment functions. These definitions are based on H&K p. 95:

(5) **Terminal Nodes (TN):** (unchanged)
If α is a terminal node, $[\![\alpha]\!]$ is specified in the lexicon.[1]

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

(7) **Functional Application (FA):**
If α is a branching node, $\{\beta, \gamma\}$ is the set of α's daughters, then, for any assignment g, if $[\![\beta]\!]^g$ is a function whose domain contains $[\![\gamma]\!]^g$, then $[\![\alpha]\!]^g = [\![\beta]\!]^g([\![\gamma]\!]^g)$.

(8) **Predicate Modification (PM):**
If α is a branching node, $\{\beta, \gamma\}$ is the set of α's daughters, then, for any assignment g, if $[\![\beta]\!]^g$ and $[\![\gamma]\!]^g$ are both of type $\langle e, t\rangle$, then $[\![\alpha]\!]^g = \lambda x \in D_e \,.\, [\![\beta]\!]^g(x) = 1$ and $[\![\gamma]\!]^g = 1$.

3 *Such that* relatives

The English expression *such that* allows us to describe complex predicates using pronouns.[2]

(9) ? This book is such$_4$ that he$_3$ bought it$_4$. ($g(3)$ = John)

Here, (9) has only one free pronoun. But the Principle of Compositionality states that $[\![S_1]\!]$ be computed based on the meaning of $[\![S_2]\!]$, which — if interpreted in isolation as in (10) — contains *two* free pronouns.

(10) He$_3$ bought it$_4$.

Idea: *Such* binds *it*, doing the work of creating a *predicate* out of the assignment-dependent sentence "John bought it."

(11) ***Such* Rule (temporary):**[3]
$[\![\text{such}_i\ \gamma]\!]^g = \lambda x_e \,.\, [\![\gamma]\!]^{[i \mapsto x]||g}$

[1] H&K proposes (p. 94) to still use $[\![\alpha]\!]$ without an assignment function superscript for *constants*, i.e. if $[\![\alpha]\!]^g$ is the same value for all assignment functions g.

[2] Unfortunately, the use of *such that* sounds "unlyrical" (Quine, 1960: §23)... but we'll ignore that here.

[3] "Such" does not have a type. That's why it can only be interpreted using the *Such* Rule.

$[i \mapsto x] \,||\, g$ is the *combination* of functions $[i \mapsto x]$ and g:

(12) **Definition: function combination**

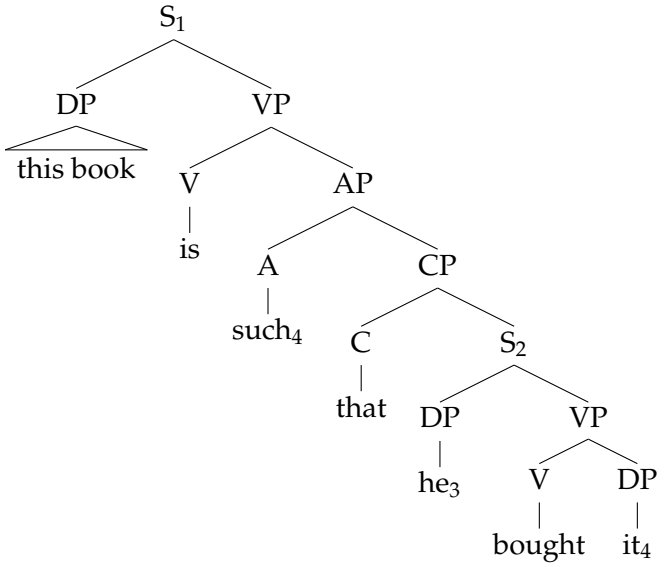
$$f \,||\, g \equiv \lambda x \,.\, \begin{cases} f(x) & \text{if } x \in \text{domain}(f) \\ g(x) & \text{otherwise} \end{cases}$$

Read "f or else g."

Warning: H&K uses $g^{x/i}$ notation for $[i \mapsto x] \,||\, g$, but I think it's confusing so I don't use it.[4]

Let's compute $[\![S_1]\!]^g$ with the following global assignment function: $g = \begin{bmatrix} 3 \mapsto \text{John} \\ 11 \mapsto \text{Tama} \end{bmatrix}$.

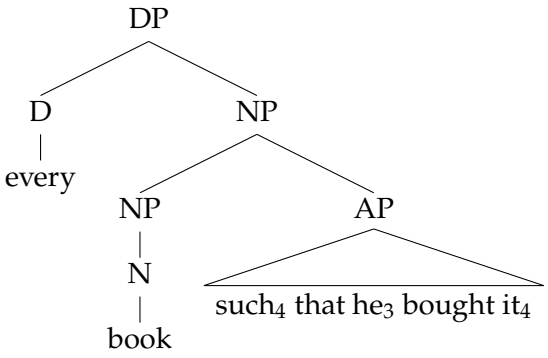
Assume $[\![\text{that}]\!] = \text{Id}$.



We can also use *such that* to construct (slightly awkward) relative clauses:

(13) ? the book such$_4$ that he$_3$ bought it$_4$

The semantics for *such* above works perfectly fine here.



[4]For one, I've also seen very similar notation "$g(x/a)$" for a function that maps x to a, which is the reverse of what H&K mean in their x/i.

4 Binding more or less than one variable

Binding multiple variables:

(14) ? This book is such$_4$ that he$_3$ bought it$_4$ and then gave it$_4$ to Sarah.

(15) ? every book such$_4$ that he$_3$ bought it$_4$ and then gave it$_4$ to Sarah

Binding no variables (vacuous binding):

(16) * This book is such$_4$ that today is Monday.

(17) * every book such$_4$ that today is Monday

The ungrammaticality of these examples shows that binding *no* variables is disallowed by the grammar. This is called *vacuous binding*.

5 Traces & Pronouns

(18) **The interpretation of movement (revised):** replaces the previous movement rule
Pick an arbitrary index i.

a. The base position of movement is replaced with a *trace* with index i: t_i.

b. A *binder index i* is adjoined right under the target position of the movement chain.

(19) **Traces and Pronouns Rule (T&P):** replaces Pronouns Rule in (2)
If α is a pronoun or trace, g is a variable assignment, and $g(i)$ is defined, then $[\![\alpha_i]\!]^g = g(i)$.

(20) **Predicate Abstraction (PA):** (H&K p. 186 version)
replaces previous rule for λ nodes in the tree and the *Such* Rule (11)[5]
Let α be a branching node with daughters β and γ, where β dominates only a numerical index i. Then, for any assignment g, $[\![\alpha]\!]^g = \lambda x\ .\ [\![\gamma]\!]^{[i \mapsto x]||g}$.

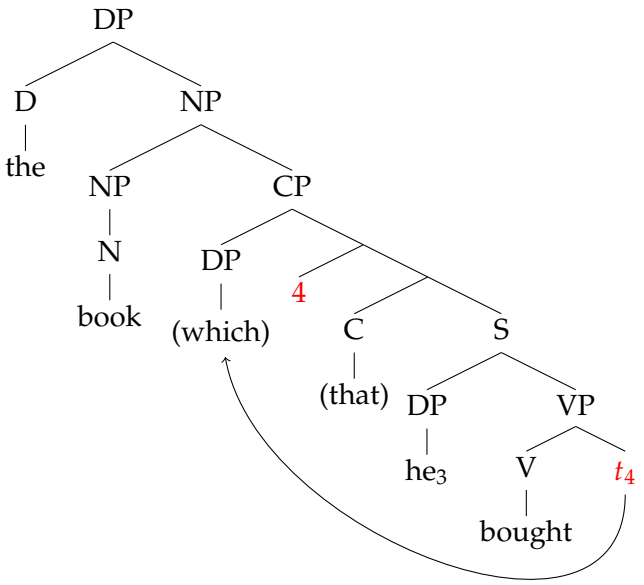
We can use T&P and PA to compute a relative clause like (21).

(21) the/every book that he$_3$ bought ___ (he_3 is free)

A motivation for thinking that traces and pronouns really are deeply related is the fact that relative clauses can simultaneously bind both traces and pronouns:

(22) the/every dog that greeted its master (Heim and Kratzer, 1998: 245)

[5] We can think of "such" as the pronunciation of a lexicalized binder index, not generated through movement.



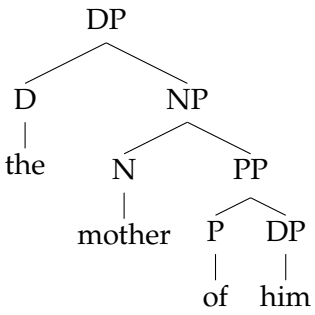
6 Variable binding

Quantifiers can also bind pronouns:

(23) Every boy loves his mother.
'Every boy is such that (x loves x's mother).'

(24) No man noticed the snake next to him. (Heim and Kratzer, 1998: 201)
'No man is such that (x noticed the snake next to x).'

Consider their LFs. Notice that the VP-internal subject hypothesis makes a contribution.
In (23), assume that "his mother" is represented as "the mother of him":



7 Variable binding and scope

The examples in this section are based on Fox 1999: 160, in turn citing Lebeaux 1995.

(25) [At least one soldier] is expected (by Napoleon) [t to die in every battle].

$^{\checkmark}\forall > \exists, ^{\checkmark}\exists > \forall$

We could imagine the scope ambiguities above could involve (a) long QR of *every battle* and/or (b) reconstruction of the subject into a lower trace position.
But notice:

(26) # [At least one soldier]$_1$ is expected by his$_1$ commander [t_1 to die in every battle].

$^{*}\forall > \exists, ^{\checkmark}\exists > \forall$

- Variable binding requires the binding quantifier to take scope above the pronoun, restricting possibilities for scope-taking. Here, this teaches us that inverse scope in (25) involves reconstruction of *at least one soldier* into its embedded clause and QR of *every battle*, rather than long QR of *every battle* alone.

References

Fox, Danny. 1999. Reconstruction, binding theory, and the interpretation of chains. *Linguistic Inquiry* 30:157–196.

Heim, Irene, and Angelika Kratzer. 1998. *Semantics in generative grammar*. Malden, Massachusetts: Blackwell.

Lebeaux, David. 1995. Where does binding theory apply? In *University of Maryland working papers in linguistics 3*, 63–88.

Quine, Willard Van Orman. 1960. *Word and object*. Cambridge.