Seth Cable
Spring 2009

A Review of Extensional Semantics, Part 2: Pronouns, Quantifiers, Quantifier Raising (QR) and Pronominal Binding [1]

1. A Quick Recap of Part 1

(1) **The Conceptual Foundations**

- Qua linguists, we want a formal model of the procedure our 'mind/brains' use to assign 'interpretations' to the structures of our languages.

- A system that seems to have a good shot of doing this is one that can derive the 'truth conditions' of sentences from the 'meanings' of their component parts.

- A system that seems to have a good shot of doing *that* is one that (i) computes the so-called 'extensions' of structures from the 'extensions' of their parts, and (ii) identifies the 'extension' of a sentence as its truth value.

- In such a system, the notion that "the meaning of a complex phrase is derived by 'combining' the meanings of its parts" is captured via the modeling of some 'meanings' as functions from objects of one type to objects of another.

- Thus, in such a system, the 'combining' of meanings is formally modeled as *function application*. (slogan, possibly false = 'all semantic composition is function application')

(2) **Some Basic Formalisms**

- Our semantic valuation function "[[.]]", which takes structures of the *object*-language to their semantic values (extensions), stated in the *meta-language*
- The 'lambda notation' for functions
- The theory of 'semantic (logical) types'

(3) **Determining the 'Meaning' of a Lexical Item L**

(a) Consider the truth-conditions of sentences in which L appears.

(b) Consider the (already established) extensions of the other lexical items in these sentences.

(c) Based on (a) and (b), develop an entry for L which would – in combination with the entries for the other words in the sentences (b) – correctly derive the truth conditions of the sentences it appears in (a).

[1] These notes summarize some of the 'highlights' of Heim & Kratzer (1998), Chapters 5 – 10.

2. Pronouns: Indices and Assignment Functions

(4) **Question**

- How should / could our system model the meanings of *pronouns?*
- What (if anything) should our semantic valuation function "[[.]]" yield as the extension of a pronoun?

Well… recall our 'procedure' in (3). We must consider the truth conditions of sentences that contain pronouns, and try to develop an entry for pronouns that captures their apparent 'contribution' to those truth conditions!

(5) **Core Data / Observations**

(a) Fact 1:
The extension of a pronoun seems to be an entity.

- *In a context where the speaker is pointing to Obama:*
  [[He smokes]] = T *iff* Obama smokes.
- Given that [[smokes]] = [λx. x smokes], this suggests [[He]] = Obama.

(b) Fact 2:
The extension of a pronoun isn't a *fixed* entity, even in a single context.

- *In a context where the speaker points first to Obama, and then to Joe.*
  [[He smokes and he doesn't smoke]] = T iff Obama smokes and Joe doesn't smoke.
- Thus, given our 'procedure' in (3), this suggests that (paradoxically) [[he]] = Obama and [[he]] = Joe

(6) **The Challenge**

a. The property in (5b) poses a serious puzzle.

b. Since "[[.]]" is by assumption a function, it cannot yield two different values for the same input.

c. Thus, it can't simultaneously be that [[he]] = Obama and [[he]] = Joe.

The standard solution/analysis comes in two steps, a syntactic one and a semantic one.

(7) **The Syntactic Step: Indices**

Pronouns are distinguished from other DPs in that they must bear an *index*.

- We can take an index to be a natural number.
- We can represent the pronoun 'bearing' the index via sub-scripting

Illustration:

Pronouns	*'Full' DPs*
he_1 , him_2 , she_{34} , it_{105} , *her	Obama, Joe, the president, *$George_3$

(8) **The Semantic Step: Interpretation of Indices**

How do we semantically interpret such 'indexed pronouns'? Here, there are three sub-steps.

a. Assignment Functions

Indices are interpreted. However, indices are not interpreted by "[[.]]" but by a special function (usually represented "g"), called an *assignment function.*

i. An *assignment function* is any function from indices (natural numbers) to entities.

Examples:

- g(1) = Dave, g(2) = Frank, g(3) = Obama, …
- h(1) = Joe, h(2) = Sarah, h(3) = John, …

b. Pairing "[[.]]" Within an Assignment Function

Our semantic valuation function "[[.]]" now always comes paired with some assignment function.

If we are pairing "**[[ ]]**" with the assignment function *g*, we write " $\mathbf{[[\ .\ ]]^g}$ "

c. Interpretation of Pronouns = Interpretation of the Index

The interpretation of a pronoun by $\mathbf{[[\ .\ ]]^g}$ is simply the interpretation assigned by *g* to the index of the pronoun.

$[[\ pro_i\ ]]^g \quad = \quad g(i)$

Special Note:
As you may recall, the rule in (8c) will also be employed for *movement-traces*.
With the assumptions in (7) and (8), we now have a means of capturing the 'puzzling' data.

(9) **The Semantic Freedom of Pronouns**

Fact to Capture:
In a context where the speaker points first to Obama, and then to Joe.
"He smokes and he dances" is T iff Obama smokes and Joe dances.

a. Central Assumptions:

(i) The sentence above has the following (abstract) syntax:
[[he_1 smokes] and [he_2 dances]]

(ii) Given the context described, the sentence above is interpreted by **[[.]]^h**, where the assignment function *h* is defined as follows:

h(1) = Obama; h(2) = Joe

> **Side-Note:**
> We still need to spell out a story clarifying how the context informally described above affects the choice of 'assignment function' we use.

b. Derivation:

To Prove:
[[[he_1 smokes] and [he_2 dances]]^h = T *iff* Obama smokes and Joe dances

(i) [[[he_1 smokes] and [he_2 dances]]]^h = T *iff* (by FA)

(ii) [[[and]]^h([[he_1 smokes]]^h)]([[he_2 dances]]^h) = T *iff* (by Lex.)

(iii) [[λp[λq [p = T and q = T]]([[he_1 smokes]]^h)]([[he_2 dances]]^h) = T *iff* (by LC) [2]

(iv) [[he_1 smokes]]^h = T and [[he_2 dances]]^h = T *iff* (by FA)

(v) [[smokes]]^h([[he_1]]^h) = T and [[dances]]^h([[he_2]]^h) = T *iff* (by Lex.)

(vi) [λx. x smokes]([[he_1]]^h) = T and [λx. x dances]([[he_2]]^h) = T *iff* (by (8c))

(vii) [λx. x smokes](h(1)) = T and [λx. x dances](h(2)) = T *iff* (by (9aii))

(viii) [λx. x smokes](Obama) = T and [λx. x dances](Joe) = T *iff* (by LC)

(ix) Obama smokes and Joe dances.

[2] Throughout the proofs in this handout, we use "Lex." to mean *lexical entry* and "LC" to mean *lambda conversion.*

3. Quantifiers and/or Quantificational Determiners

(10) **Question**

- How should / could our system model the meanings of *quantificational determiners* (*e.g.* "every", "some", "three", "many", "most", *etc.*)?

- What (if anything) should our semantic evaluation function **[[.]]^{g}** yield as the extension of such determiners?

Well… recall our 'procedure' in (3). We must consider the truth conditions of sentences that contain such 'quantificational determiners', and try to develop an entry for them that captures their apparent 'contribution' to those truth conditions!

Side-Note:
Since these are just review notes, I'll confine my attention to the quantificational determiner "every". See your notes from last semester (or Heim & Kratzer (1998)) for treatment of "some", "most", "few", etc.

(11) **Starting Observations**

a. Truth Conditions of (Simple) Sentences Containing "Every"

[[Every boy smokes]] = T *iff* for all x, if x is a boy, then x smokes.

b. Background Assumptions Regarding Other Lexical Items

(i) [[smokes]] = [λx. x smokes]
(ii) [[boy]] = [λx. x is a boy]

Side-Note: When the assignment function "g" isn't important, we eliminate it from our notation.

(12) **Initial Puzzle**

How do we combine anything with the extensions in (11b) to get the truth conditions in (11a)??

Plan of Action: Let's proceed incrementally.

(a) Let's first come up with an extension for "every boy" that will be able to combine with [[smokes]] to yield the truth-conditions in (11a).

(b) Once we've done that, let's come up with an extension for "every" that will be able to combine with [[boy]] to yield the extension we devised in step (12a)!

(13) **Step 1: The Extension of "Every Boy"**

a. Starting Observations

(i) For various conceptual reasons, it's seems wrong to treat the extension of "every boy" as being a type *e* entity that is argument to the extension of "smokes"…
(*what kind of entity is 'every boy'*?)

(ii) *So maybe we should think of the extension of "every boy" as* ***itself*** *being a function that takes the extension of "smokes" as argument.*

b. The Proposal

[[every boy]] = $\lambda P_{<et>}$. for all x, if x is a boy, then P(x) = T

The function that takes a function P *of type* <et> *and yields:*
T *iff* for all x, if x is a boy, then P(x) is true.

c. Illustrative Derivation

To Prove:
[[Every boy smokes]] = T *iff* for all x, if x is a boy, then x smokes.

(i) [[Every boy smokes]] = T *iff* (by FA)

(ii) [[every boy]] ([[smokes]]) = T *iff* (by (13b))

(iii) [$\lambda P_{<et>}$. for all x, if x is a boy, then P(x) = T]([[smokes]]) = T *iff* (by LC)

(iv) for all x, if x is a boy, then [[smokes]](x) = T *iff* (by Lex.)

(v) for all x, if x is a boy, then [λy. y smokes](x) = T *iff* (by LC)

(vi) for all x, if x is a boy, then x smokes

Yeah! Step 1 Complete!

Side-Note:

Following our proposal in (13b), quantificational DPs are of type <et, t>.

A general term for this kind of function (*i.e.* of type <et, t>) is 'generalized quantifier'.

So, if we assume that (13b) is the extension of "every boy", our system will derive the targeted truth-conditions in (11a)…

…now, let's figure out an extension for "every" that, when combined with the extension for "boy" (in (11b)) will give us our hypothesized extension for "every boy" in (13b)!!

(14) **Step 2: The Extension of "Every"**

a. Starting Observation

Just as we concluded earlier (13aii), perhaps we should view the extension of "every" as some complex higher order function, which takes the extension of "boy" as its *argument.*

b. The Proposal

[[every]] = $\lambda Q_{<et>}$ [$\lambda P_{<et>}$. for all x, if Q(x) = T, then P(x) = T]

The function that takes a function Q *of type* <et> *and yields:*
The function that takes a function P *of type* <et> *and yields:*
T *iff* for all x, if Q(x) is T, then P(x) is T

c. Illustrative Derivation

To Prove:
[[every boy]] = $\lambda P_{<et>}$. for all x, if x is a boy, then P(x) = T

(i) [[every boy]] = (by FA)

(ii) [[every]]([[boy]]) = (by (14b))

(iii) [$\lambda Q_{<et>}$ [$\lambda P_{<et>}$. for all x, if Q(x) = T, then P(x) = T]]([[boy]]) = (by LC)

(iv) [$\lambda P_{<et>}$. for all x, if [[boy]](x) = T, then P(x) = T]] = (by Lex.)

(v) [$\lambda P_{<et>}$. for all x, if [λy. y is a boy](x) = T, then P(x) = T]] = (by LC)

(vi) [$\lambda P_{<et>}$. for all x, if x is a boy, then P(x) = T]]

Yeah! Step 2 Complete!

(15) **Summary: The Semantics of Quantificational Determiners**

Quantificational determiners like "every" are of type <et < et, t>>

- First, they combine with the extension of their NP complements, which is <et>
- Then, they combine with another <et> function. In the example above, this was the VP which was sister to the quantificational subject.
- Once they combine with their NP complement and another predicate (*e.g.* VP), they produce a truth-value.

(16) **Additional Technology**

In the subsequent discussion, I will assume the following semantics for the indefinite Ds "some" and "a" (which are also taken to be of type <et <et, t>>):

[[some / a]] = $\lambda Q_{<et>}$ [$\lambda P_{<et>}$. there is an x, such that Q(x) = T and P(x) = T]

---

4. Quantifier Raising (QR): Its Syntax and Semantics

(17) **A Puzzle**

The sentence in (a) below allows an interpretation where it has the truth conditions in (b).

a. A boy dates every girl.

b. For all y, if y is a girl, then there is an x, such that x is a boy and x dates y

<u>In a Picture:</u>
Sentence (17a) can be interpreted as T when the following facts hold:
Boys = { Dave, John, Bill }
Girls = { Sarah, Lauren, Ronda }
Dave dates Sarah. Bill dates Ronda. Dave dates Lauren.

(18) **The Two-Part Challenge**

a. How can we derive (b) as a possible reading of (a)?

b. **Given our semantics for "every", "every girl" should <u>not</u> be able to appear in the object position of "dates".**

"every girl" is of type <et, t> **"dates" is of type <e, et>**

(19) **The Solution (or rather, *One* Solution)**

Let us assume that the quantificational object "every girl" can undergo a special *covert movement* called Quantifier Raising (QR).

- 'After' you pronounce the sentence, the quantificational object can undergo movement to a position above the quantificational subject.
- Semantics interprets this later, post-pronunciation structure ("L(ogical) F(orm)")

If we play our cards right, this can provide a solution to both parts of the 'two-part' challenge in (18)!

Side-Note:

- As you most likely recall, there are *many* other solutions to the challenges in (18).
- However, all the work we will be reading this semester will be assuming the 'QR analysis' of quantifier scope, and so our review session here need only cover this analysis.

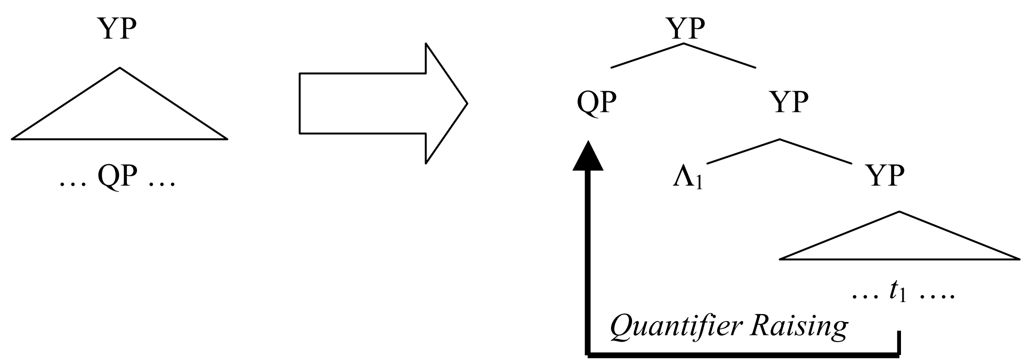
(20) **The Syntax of Movement**

So-called 'movement' of a phrase QP to a position sister to YP results in the following:

(a) a *trace*, which is a pronoun of type *e*, appears in the initial position of QP

(b) An object-language 'lambda operator' is inserted as an adjunct to YP

(c) The inserted 'lambda operator' is co-indexed with the trace left by QP



Temporary Notational Convention:
In order to avoid confusion, I'll (at first) represent object-language (English) lambda-operators using upper-case lambdas.

(21) **Towards The Semantics of Movement**

Assuming that our syntax (along with QR) generates structures like those in (20), *how does our semantics then interpret them?*

(a) The "QP" will be interpreted according to the rules laid out in Section 3

(b) *But what about the YP with the 'lambda operator' adjoined to it?*

- (Given the notation that we are using – lambdas – we are obviously aiming for these to come out as lambda-expressions of some sort)
- And, since these YPs are sister to the QPs (generalized quantifiers), they should be of type <e t>
- So, we want for a structure of the form $[_{YP}\ \Lambda_1\ [_{YP} \ldots t_1 \ldots]\ ]$ to have as its extension an <et> function (characterized in lambda notation)

(22) **Some Additional Notation**

" $g(\,i \rightarrow a\,)$ " = *The assignment function which is just like* g, ***except that*** *it yields the value* a *for the index* i.

<u>Example:</u>

a. h = { <1, Bill>, <2, Frank>, <3, Tom>, <4, Andy>, …}
b. h(3→John) = { <1, Bill>, <2, Frank>, <3, John>, <4, Andy>, …}

(23) **The Semantics of Movement**

$$[\![\ [_{XP}\ \Lambda_i\ [_{YP} \ldots t_i \ldots]]\ ]\!]^g \quad = \quad \lambda x\ .\ [\![\ [_{YP} \ldots t_i \ldots]\ ]\!]^{g(i \rightarrow x)}$$

"For any assignment function *g*:
If XP is a structure consisting of two daughters – an object language lambda operator with index *i* and a phrase YP, then the extension of XP is:
The function which given an entity *a* yields T iff
YP is true under the assignment function *g*(i→*a*)"

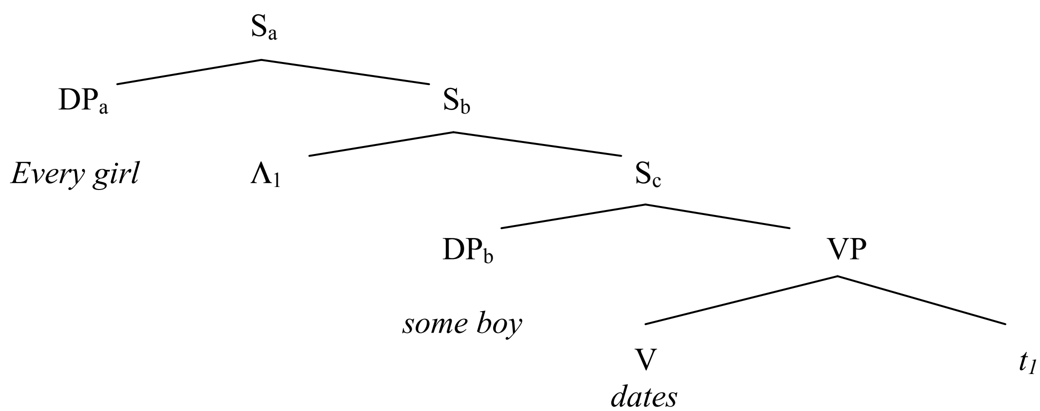
OR

"… The function defined by a lambda-operator 'λx' taking scope over
The interpretation of YP under the assignment function *g*(i→x)

Before we try to understand in detail what the rule in (23) is saying exactly, let's see how it works in action to derive the targeted truth conditions in (17b).

(24) **Derivation of Reading (17b) for Sentence (17a)**

a. Syntax:
Under reading (17b), sentence (17a) is assumed to have the following syntactic representation.



b. Semantics:
$[[S_a]]^g = T$ *iff* for all y, if y is a girl, there is an x, such that x is a boy & x dates y

(i) $[[S_a]]^g = T$ *iff* (by FA)

(ii) $[[\text{every girl}]]^g ([[S_b]]^g) = T$ *iff* (by FA, Lex.)

(iii) $[\lambda P_{<et>}$. for all y, if y is a girl, then $P(y) = T]([[S_b]]^g) = T$ *iff* (by LC)

(iv) For all y, if y is a girl, then $[[S_b]]^g(y) = T$ *iff* (by (23))

(v) For all y, if y is a girl, then $[\lambda z . [[S_c]]^{g(1\rightarrow z)}](y) = T$ *iff* (by FA)

(vi) For all y… $[\lambda z. [[\text{some boy}]]^{g(1\rightarrow z)}([[VP]]^{g(1\rightarrow z)})](y) = T$ *iff* (by FA, Lex.)

(vii) For all y… $[\lambda z. [\lambda P_{<et>}$. there is an x,
such that x is a boy and $P(x) = T]([[VP]]^{g(1\rightarrow z)})](y) = T$ *iff* (by LC)

(viii) For all y… $[\lambda z.$ there is an x,
such that x is a boy and $[[VP]]^{g(1\rightarrow z)}(x) =T](y)$ *iff* (by FA)

(ix) For all y… $[\lambda z.$ there is an x,
such that x is a boy and $[[[\text{dates}]]^{g(1\rightarrow z)}([[t_1]]^{g(1\rightarrow z)})](x) =T](y)$ *iff* (by Lex.)

(x) For all y… $[\lambda s. \lambda t$. t dates $s]([[t_1]]^{g(1\rightarrow z)})](x) =T](y)$ *iff* (by (8c))

Continued….

(xi) For all y… [λs. λt . t dates s](g(1→z)(1))](x) = T](y) *iff* (by def. of '*g*(1→z)')

(xii) For all y… [λs. λt . t dates s](z)](x) = T](y) *iff* (by LC)

(xiii) For all y… [λz. there is an x,
such that x is a boy and [λt . t dates z](x) = T](y) *iff* (by LC)

(xix) For all y, if y is a girl, then [λz. there is an x,
such that x is a boy and x dates z](y) *iff* (by LC)

(xx) **For all y, if y is a girl, then there is an x, such that x is a boy and x dates y**

4.1 Some Discussion

(25) **How the 'QR' Analysis Works**

Here are the answers our QR-based system provides to the 'puzzles' we started off with:

(a) We are able to derive the (so-called 'wide-scope') reading in (17b) by moving the quantificational object (or 'QR-ing it') to a position that c-commands the subject.

- Given our semantics for quantificational DPs, such movement would predict that the quantificational object will have *scope over* the quantificational subject.

(b) By assuming that movement of the QP leaves a 'trace', and that this trace is a pronoun of type *e*, our system predicts that the transitive verb in sentences like (17a) should still be able to compose with the material in its object position.

(26) **On the Interpretation of Traces**

When the trace t_1 above is interpreted by the assignment function 'g(1→z)', what *exactly* does the trace 'denote' or 'refer to'?

a. *A variable?*
No! In our *meta*-language 'z' doesn't *refer* to a variable, but *is* a variable. It is a variable over entities.

b. *An entity, then?*
Well, no again! In our semantic system, any meta-language variable (like *z*) will always be *bound* by meta-language lambda operators.

c. *So, what, then?*
Under this assignment, the trace (pronoun) doesn't, properly speaking, denote/refer to *anything*. Rather, it *functions* as a bound-variable.

(27) **On the Use of Syntactic 'Movement' to Determine Scope**

The *sine qua non* of the QR analysis is the following claim:

a. The Existence of Covert Movement

Let S be a sentence whose *surface* structure contains two quantificational DPs – X and Y – which are such that X c-commands Y:

If S is interpreted so that Y takes X within its quantificational scope, then the LF structure of S is one in which Y has undergone **movement** to a position c-commanding X

As you may recall, a classic argument in favor of the claim in (27a) is the generalization in (27b).

b. The Scope-Movement Generalization

Let S be a sentence whose *surface* structure contains two quantificational DPs – X and Y – which are such that X c-commands Y:

If a **movement-island** intervenes between X and Y, then S cannot be interpreted so that Y takes X within its quantificational scope.

c. Some Illustrative Examples of Generalization (27b)

Relative Clause Islands

(i) Most boys that saw Frank yelled about it.
(ii) * Who did most boys that saw ___ yell about it?

(iii) Most boys that saw every girl yelled about it.
(iv) =/= *For every x, if x is a girl, then most boys that saw x yelled about it.*

However, as you *also* may recall (and as we'll see in a couple weeks), the actual facts regarding licit quantifier-scopes are *much* more complicated and confusing than this.

A full picture of the facts may well challenge the generalization in (27b), and so weaken the force of the data paradigm in (27c)….

5. Pronominal Binding

The final subject to 'refresh ourselves' on is the syntax and semantics of *bound* pronouns. Interestingly, the tools we've developed above are already enough to capture the basic facts!

(28) **Central Observation**

The (non-parenthetical) sentence in (28a) is ambiguous, and permits the readings paraphrased in (28b) and (28c).

a. (Tom saw a car that pleased him. In fact...) Every boy saw a car that pleased him.

b. Referential Reading
For all x, if x is a boy, then there is a y such that y is a car and y pleased **Tom**, and x saw y.

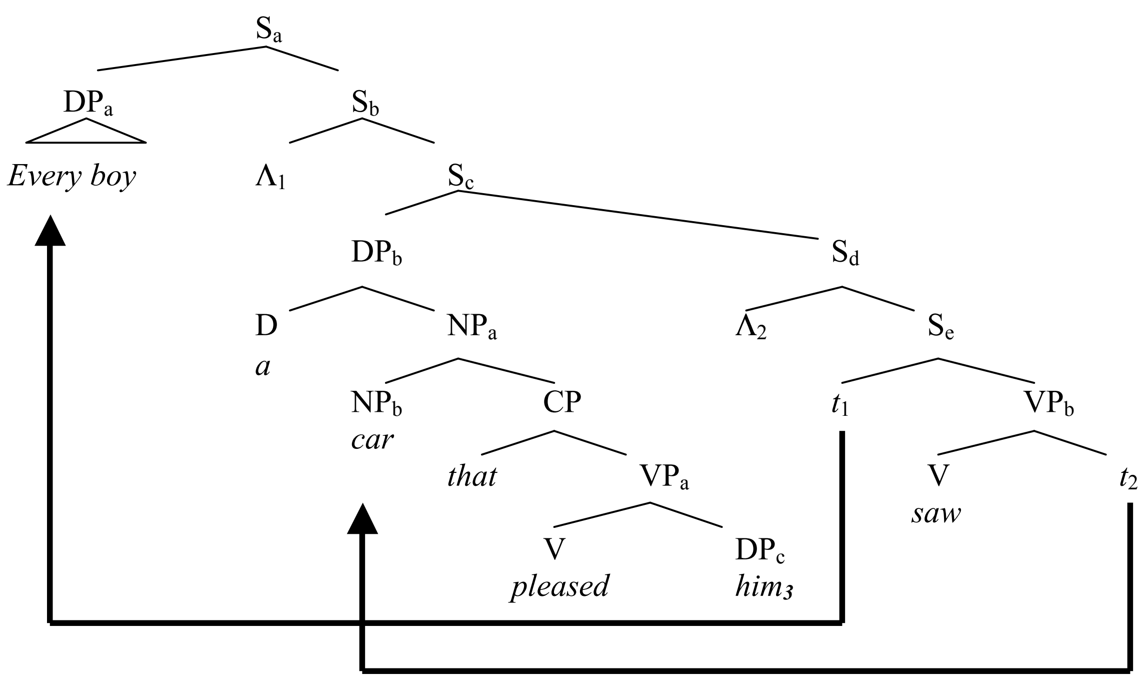
c. Bound Reading
For all x, if x is a boy, then there is a y such that y is a car and y pleased **x**, and x saw y.

(29) **Central Claim**

Our semantics for pronouns, our semantics for quantifiers, and our semantics for movement structures is able to capture the existence of both readings for (28a)

(30) **The Referential Reading: Syntax**

Under the 'referential reading' in (28b), the structure of (28a) at LF is the following:



(31) **The Referential Reading: Semantics**

Let *h* be an assignment function with the following property: h(3) = Tom
To Prove: $[[S_a]]^h = T$ *iff* for all x, if x is a boy, then there is a y such that y is a car and y pleased **Tom**, and x saw y.

(i) $[[S_a]]^h = T$ *iff* (by FA)

(ii) $[[\text{Every boy}]]^h ([[S_b]]^h) = T$ *iff* (by Lex., FA)

(iii) $[\lambda P_{<et>}.$ for all x, if x is a boy, then $P(x) = T]([[S_b]]^h) = T$ *iff* (by LC)

(iv) for all x, if x is a boy, then $[[S_b]]^h(x) = T$ *iff* (by (23))

(v) for all x, if x is a boy, then $[\lambda z. [[S_c]]^{h(1\rightarrow z)}](x) = T$ *iff* (by FA)

(vi) for all x, … $[\lambda z. [[DP_b]]^{h(1\rightarrow z)}([[S_d]]^{h(1\rightarrow z)})](x) = T$ *iff* (by (23))

(vii) for all x, … $([\lambda s. [[S_e]]^{h(1\rightarrow z;\ 2\rightarrow s)}])](x) = T$ *iff* (by FA)

(viii) for all x, … $([\lambda s. [[VP_b]]^{h(1\rightarrow z;\ 2\rightarrow s)}([[t_1]]^{h(1\rightarrow z;\ 2\rightarrow s)})])](x) = T$ *iff* (by (8c))

(ix) for all x, … $([\lambda s. [[VP_b]]^{h(1\rightarrow z;\ 2\rightarrow s)}(z)])](x) = T$ *iff* (by FA, Lex.)

(x) for all x, … $([\lambda s. [\lambda t.\lambda u.\ u \text{ saw } t.([[t_2]]^{h(1\rightarrow z;\ 2\rightarrow s)})](z)])](x) = T$ *iff* (by (8c))

(xi) for all x, … $([\lambda s. [\lambda t.\lambda u.\ u \text{ saw } t.(s)](z)])](x) = T$ *iff* (by LC)

(xii) for all x, … $[\lambda z. [[DP_b]]^{h(1\rightarrow z)}([\lambda s.\ z \text{ saw } s])](x) = T$ *iff* (by FA)

(xiii) for all x, … $[\lambda z. [\ [[a]]^{h(1\rightarrow z)}([[NP_a]]^{h(1\rightarrow z)})\ ]([\lambda s.\ z \text{ saw } s])](x) = T$ *iff* (by PM)

(xiv) for all x, … $(\lambda v. [[NP_b]]^{h(1\rightarrow z)}(v) = T\ \&\ [[CP]]^{h(1\rightarrow z)}(v) = T)$ … = T *iff* (by Lex.)

(xv) for all x, … $(\lambda v. [\lambda u.u \text{ is a car}](v) = T\ \&\ [[CP]]^{h(1\rightarrow z)}(v) = T)$ … = T *iff* (by LC)

(xvi) for all x, … $(\lambda v.\ v \text{ is a car}\ \&\ [[CP]]^{h(1\rightarrow z)}(v) = T)$ … = T *iff* (by Special Assump.)

(xvii) for all x, … $(\lambda v.\ v \text{ is a car}\ \&\ [[VP_a]]^{h(1\rightarrow z)}(v) = T)$ … = T *iff* (by FA)

(xviii) for all x, … $[\ [[\text{pleased}]]^{h(1\rightarrow z)}([[\text{him}_3]]^{h(1\rightarrow z)})\ ](v) = T)$ …= T *iff* (by Lex.)

(xix) for all x, … $[\ \lambda q\ \lambda r.\ r \text{ pleased } q\ ]([[\text{him}_3]]^{h(1\rightarrow z)})\ ](v) = T)$ …= T *iff* (by (8c), def. of *h*)

(xx) for all x, … [λq λr. r pleased q](Tom)](v) = T) …= T *iff* (by LC)

(xxi) for all x, … [λr. r pleased Tom](v) = T) …= T *iff* (by LC)

(xxii) for all x, … [λz. [$[[a]]^{h(1\rightarrow z)}$(λv. v is a car & v pleased Tom)] … = T *iff* (by Lex.)

(xxiii) for all x, … [λPλQ. there is a y such that P(y) = T and Q(y) = T] … = T *iff* (by LC)

(xxiv) for … [λQ. there is a y such that [λv. v is a car & v pleased Tom](y) … =T *iff* (by LC)

(xxv) … [λQ. there is a y such that y is a car & y pleased Tom and Q(y) = T] …= T *iff* (by LC)

(xxvi) for all x, … [there is a y … and [λs. z saw s](y) = T]](x) = T *iff* (by LC)

(xxvii) for all x, if x is a boy, then [λz. there is a y … and z saw y]](x) = T *iff* (by LC)

(xxviii) **for all x, if x is a boy then there is a y such that y is a car & y pleased Tom & x saw y.**

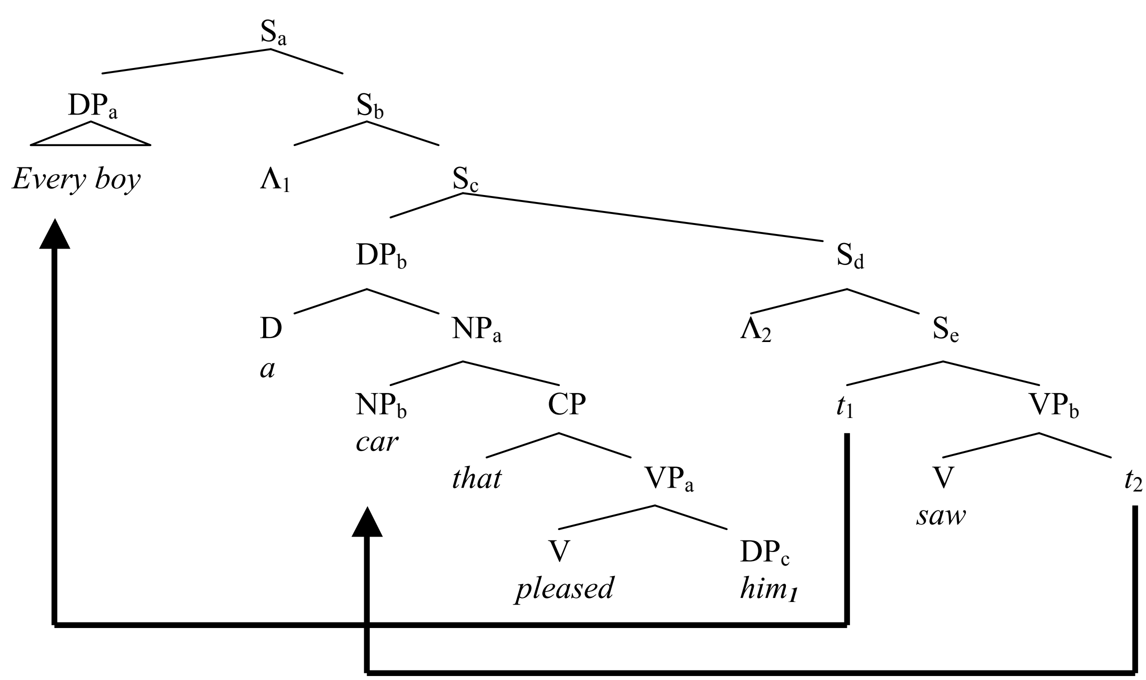
(32) **Summary**

Our system derives the 'referential reading' of (28a) in (28b) if the following holds:

- The pronoun 'him' receives an index *distinct from* the indices born by the (meta-language) lambda-operators that c-command the pronoun.
- We interpret the entire sentence using an assignment function where the index of the pronoun is sent to the understood referent, Tom.

(33) **The Bound Reading: Syntax**

Under the 'bound reading' in (28c), the structure of (28a) at LF is the following:



(34) **The Bound Reading: Semantics**

To Prove: $[[\ S_a\ ]]^h = T$ *iff* for all x, if x is a boy, then there is a y such that y is a car and y pleased **x**, and x saw y.

(i) $[[\ S_a\ ]]^h = T$ *iff* (by FA)

(ii) $[[\text{ Every boy}]]^h ([[S_b]]^h) = T$ *iff* (by Lex., FA)

(iii) $[\lambda P_{<et>}.$ for all x, if x is a boy, then $P(x) = T]([[S_b]]^h) = T$ *iff* (by LC)

(iv) for all x, if x is a boy, then $[[S_b]]^h(x) = T$ *iff* (by (23))

(v) for all x, if x is a boy, then $[\lambda z.\ [[S_c]]^{h(1\rightarrow z)}](x) = T$ *iff* (by FA)

(vi) for all x, … $[\lambda z.\ [[DP_b]]^{h(1\rightarrow z)}([[S_d]]^{h(1\rightarrow z)})](x) = T$ *iff* (by (23))

(vii) for all x, … $([\lambda s.\ [[S_e]]^{h(1\rightarrow z;\ 2\rightarrow s)}])](x) = T$ *iff* (by FA)

(viii) for all x, … $([\lambda s.\ [[VP_b]]^{h(1\rightarrow z;\ 2\rightarrow s)}([[t_1]]^{h(1\rightarrow z;\ 2\rightarrow s)})])](x) = T$ *iff* (by (8c))

(ix) for all x, … $([\lambda s.\ [[VP_b]]^{h(1\rightarrow z;\ 2\rightarrow s)}(z)])](x) = T$ *iff* (by FA, Lex)

(x) for all x, … $([\lambda s.\ [\lambda t.\lambda u.\ u \text{ saw } t.([[t_2]]^{h(1\rightarrow z;\ 2\rightarrow s)})](z)])](x) = T$ *iff* (by (8c))

(xi) for all x, … $([\lambda s.\ [\lambda t.\lambda u.\ u \text{ saw } t.(s)](z)])](x) = T$ *iff* (by LC)

(xii) for all x, … $[\lambda z.\ [[DP_b]]^{h(1\rightarrow z)}([\lambda s.\ z \text{ saw } s])](x) = T$ *iff* (by FA)

(xiii) for all x, … $[\lambda z.\ [\ [[a]]^{h(1\rightarrow z)}([[NP_a]]^{h(1\rightarrow z)})\ ]([\lambda s.\ z \text{ saw } s])](x) = T$ *iff* (by PM)

(xiv) for all x, … $(\lambda v.\ [[NP_b]]^{h(1\rightarrow z)}(v) = T\ \&\ [[CP]]^{h(1\rightarrow z)}(v) = T)$ … $= T$ *iff* (by Lex.)

(xv) for all x, … $(\lambda v.\ [\lambda u.u \text{ is a car}](v) = T\ \&\ [[CP]]^{h(1\rightarrow z)}(v) = T)$ … $= T$ *iff* (by LC)

(xvi) for all x, … $(\lambda v.\ v \text{ is a car}\ \&\ [[CP]]^{h(1\rightarrow z)}(v) = T)$ … $= T$ *iff* (by Special Assump.)

(xvii) for all x, … $(\lambda v.\ v \text{ is a car}\ \&\ [[VP_a]]^{h(1\rightarrow z)}(v) = T)$ … $= T$ *iff* (by FA)

(xviii) for all x, … $[\ [[\text{pleased}]]^{h(1\rightarrow z)}([[\text{him}_1]]^{h(1\rightarrow z)})\ ](v) = T)$ …$= T$ *iff* (by Lex)

(xix) for all x, … $[\ \lambda q\ \lambda r.\ r \text{ pleased } q\ ]([[\text{him}_1]]^{h(1\rightarrow z)})\ ](v) = T)$ …$= T$ *iff* (by (8c))

(xx) for all x, … $[\ \lambda q\ \lambda r.\ r \text{ pleased } q\ ](z)\ ](v) = T)$ …$= T$ *iff* **(by LC)**

(xxi) for all x, … [λr. r pleased z](v) = T) …= T *iff* (by LC)

(xxii) for all x, … [λz. [$[[a]]^{h(1\rightarrow z)}$(λv. v is a car & v pleased z)] … = T *iff* (by Lex.)

(xxiii) for all x, … [λPλQ. there is a y such that P(y) = T and Q(y) = T] … = T *iff* (by LC)

(xxiv) for … [λQ. there is a y such that [λv. v is a car & v pleased z](y) … =T *iff* (by LC)

(xxv) … [λQ. there is a y such that y is a car & y pleased z and Q(y) = T] …= T *iff* (by LC)

(xxvi) for all x, … [there is a y … and [λs. z saw s](y) = T]](x) = T *iff* (by LC)

(xxvii) for all x, if x is a boy, then [λz. there is a y … and z saw y]](x) = T *iff* (by LC)

(xxviii) **for all x, if x is a boy then there is a y such that y is a car & y pleased x & x saw y.**

(35) **Summary**

Our system derives the 'bound reading' of (28a) in (28c) if the following holds:

- The pronoun 'him' receives *the same* index as is born by the (meta-language) lambda-operator that is sister to the quantificational DP.
- *Regardless of the assignment function we choose*, such a structure will receive the 'bound reading' in (28c)

(36) **Some Additional Points**

(a) Strictly speaking, in this system, pronouns are bound by *lambda operators*. It is only in a more indirect and derivative sense that pronouns are 'bound' by DPs themselves. (See the discussion of 'binding' in Heim & Kratzer 1998)

(b) However, our system does still captures the common notion that a pronoun can only be 'bound' by a DP (in this derived sense) if that DP *c-commands* the pronoun. (See the discussion of 'binding' in Heim & Kratzer 1998)