An Algebraic Approach to Quantification and Lambda Abstraction: Applications to the Analysis of English

(1) **Ingredients on the Table**
a. A logical language with both quantification and lambda abstraction (Politics+λ)
b. An interpretation $<B, G_\gamma, f>_{\gamma \in \{Concat, Not, And, Or, If, \exists, \forall, \lambda\}}$ for Politics+λ

(2) **Goals for This Unit**

a. Develop a fragment of English that contains quantificational NPs.

b. Develop a translation base from that fragment to Politics+λ

c. Examine the interpretation thereby induced for the fragment of English.

Note: The system developed here is a blend of the ones Montague develops in UG and PTQ. To my knowledge, this system appears nowhere else in the literature on MG.

1. Towards a New Fragment of English: Quantificational NPs in Subject Position

In this section, we will work out at a relatively informal level how we should structure the English fragment and its translation into Politics+λ

(3) **Proximal Goal**
Expand our fragment of English so it includes expressions like: *some man*, *every man*.

(4) a. Easy Question: What should the category of these new expressions be?

b. Easy Answer:
Well, their external syntactic behavior is just like the expressions *Barack, Mitt, Michelle*. Thus, if the latter are NPs, then so should be *some man* and *every man*.

(5) **First Steps Towards Implementation**
We're not ready to officially write out this new fragment of English, but it seems that it should include the following pieces.

a. The New Category Labels:

(i)	TV	'transitive verb'
(ii)	IV	'intransitive verb' (and 'VPs')
(iii)	S	'sentence'
(iv)	**T**	**'term' (formerly 'NP')**
(v)	**CN**	**'common noun' (that is, the 'Ns')**

b. <u>The Basic Expressions</u>

(i) X_{TV} = { < *loves*, ∅ > }
(ii) X_{IV} = { < *smokes*, ∅ > }
(iii) $\mathbf{X_T}$ = { < *Barack*, ∅ >, < *Mitt*, ∅ >, < *Michelle*, ∅ > }
(iv) $\mathbf{X_{CN}}$ = { < *man*, ∅ >, < *president*, ∅ > }
(v) X_S = ∅

c. <u>The Syntactic Operations</u>
In addition to the operations we introduced for DME, let's add the operations K_{Some} and K_{Every}.

- In the definitions below, α and β are trees whose root nodes are ordered pairs. In addition α' and β' are the first members of the root nodes of α and β.

(i) $K_{\text{Some}}(\alpha) = \begin{array}{c} < \textit{some } \alpha' , \text{Some} > \\ | \\ \alpha \end{array}$

(ii) $K_{\text{Every}}(\alpha) = \begin{array}{c} < \textit{every } \alpha' , \text{Every} > \\ | \\ \alpha \end{array}$

d. <u>The Syntactic Rules</u>
Let's add the following rules to the set of syntactic rules S_E we had for DME.

(i) < K_{Some} , < CN >, T >
'The result of applying K_{Some} to an expression of category CN is an expression of category T.'

(ii) < K_{Every} , < CN >, T >
'The result of applying K_{Every} to an expression of category CN is an expression of category T.'

e. <u>Illustration:</u> The category C_T includes the following trees:

< *Barack*, ∅ > < *Mitt*, ∅ > < *Michelle*, ∅ >

$\begin{array}{c} < \textit{every man}, \text{Every} > \\ | \\ < \textit{man}, \varnothing > \end{array}$ $\begin{array}{c} < \textit{every president}, \text{Every} > \\ | \\ < \textit{president}, \varnothing > \end{array}$

$\begin{array}{c} < \textit{some man}, \text{Some} > \\ | \\ < \textit{man}, \varnothing > \end{array}$ $\begin{array}{c} < \textit{some president}, \text{Some} > \\ | \\ < \textit{president}, \varnothing > \end{array}$

(6) **Handy Abbreviation**
Since it's clear from context here that the expressions of our fragment are *trees*, I'll make use of the convention below to save space:
<u>*expression*</u> = A tree whose root node is < *expression* , Index >

(7) **Difficult Question:**
What type of expressions in Politics+λ should our (quantificational) terms translate as?

(8) **Towards the Answer (Linguistics 610)**

- We don't want the meaning of <u>some man</u> or <u>every man</u> end up being of type e

- **Rather, we want it to be a function of type <<e,t>,t> (Generalized Quantifier)**

  h°*k*(<u>some man</u>) =
  the function whose domain is $D_{<et>,E}$, and for every *f* in $D_{<et>,E}$ maps *f* to 1 iff
  there is an x in $D_{e,E}$ such that x is a man and *f*(x) = 1

  h°*k*(<u>every man</u>) =
  the function whose domain is $D_{<et>,E}$, and for every *f* in $D_{<et>,E}$ maps *f* to 1 iff
  for all x in $D_{e,E}$, if x is a man then *f*(x) = 1

- Therefore, we'd like our translation function *k* to achieve the following mappings:

  k(<u>some man</u>) = $(\lambda P_0\ \exists x_0\ (\ (\mathbf{man'}\ x_0)\ \&\ (P_0\ x_0)\ )\ )$

  k(<u>every man</u>) = $(\lambda P_0\ \forall x_0\ (\ (\mathbf{man'}\ x_0) \rightarrow (P_0\ x_0)\ )\ )$

(9) **A Challenge Appears!**

- Thus, we want our translation function *k* to map <u>some man</u> and <u>every man</u> to expressions in Politics+λ of type <<e,t>,t>

- But, recall that under our theory of translation, every expression of the same category in English must be translated to an expression of the same type in Politics+λ

- **Thus, the goal in (8) entails that *every term of our language* – even <u>*Barack*</u>, <u>*Michelle*</u>, and <u>*Mitt*</u> – must be translated as <<et>,t> formulae.**

(10) **The Solution to the Challenge (Linguistics 610)**
The proper names Barack, Michelle, and Mitt will receive the translations below:

a. k(< *Barack*, ∅ >) = (λP_0 (P_0 **barack'**))
b. k(< *Mitt*, ∅ >) = (λP_0 (P_0 **mitt'**))
c. k(< *Michelle*, ∅ >) = (λP_0 (P_0 **michelle'**))

That is, the name (e.g.) Barack will end up translated as an expression denoting the characteristic function of the set of <e,t> functions that map Barack to 1.

(11) **Immediate Problem**
In our translation base mapping Mini-English to Politics-NoQ, the syntactic operation $K_{\text{Merge-S}}$ corresponds to the polynomial operation F_{Concat}<$\text{Id}_{2,2}$, $\text{Id}_{1,1}$>.

- Given the translations in (8)-(10), if we maintain that correspondence, we'll end up mapping meaningful expressions of Mini-English to syntactic garbage in Politics+λ.

k(Barack smokes) =

k($K_{\text{Merge-S}}$(< *Barack*, ∅ >, < *smokes*, ∅ >)) =

F_{Concat}<$\text{Id}_{2,2}$, $\text{Id}_{1,1}$>(k(< *Barack*, ∅ >), k(< *smokes*, ∅ >)) =

F_{Concat}<$\text{Id}_{2,2}$, $\text{Id}_{1,1}$>((λP_0 (P_0 **barack'**)) , **smokes'**) =

(**smokes'** (λP_0 (P_0 **barack'**))) = **syntactic garbage!**

(12) **Potential Solution**
For our new fragment of English, in the translation to Politics+λ, $K_{\text{Merge-S}}$ could now just correspond to F_{Concat}. That would yield the right results!

k(Barack smokes) =

k($K_{\text{Merge-S}}$(< *Barack*, ∅ >, < *smokes*, ∅ >)) =

F_{Concat}(k(< *Barack*, ∅ >), k(< *smokes*, ∅ >)) = (by (10))

F_{Concat}((λP_0 (P_0 **barack'**)) , **smokes'**) =

((λP_0 (P_0 **barack'**)) **smokes'**) ⇔ (by lambda-conversion)

(smokes' barack')

Our solution in (12) would also get the right results for sentences whose subjects are quantificational terms!...

(13) **Illustration: *<u>Every man smokes</u>***

k(*<u>every man smokes</u>*) =

k($K_{\text{Merge-S}}$(*<u>every man</u>* , < *smokes*, ∅ >)) =

F_{Concat}(*k*(*<u>every man</u>*), *k*(< *smokes*, ∅ >)) = (by (8))

F_{Concat}(($\lambda P_0\ \forall x_0$ ((**man'** x_0) → ($P_0\ x_0$))) , **smokes'**) =

(($\lambda P_0\ \forall x_0$ ((**man'** x_0) → ($P_0\ x_0$))) **smokes'**) ⇔ (by lambda-conversion)

$\forall x_0$ ((**man'** x_0) → (**smokes'** x_0))

(14) **Illustration: *<u>Some man smokes</u>***

k(*<u>some man smokes</u>*) =
k($K_{\text{Merge-S}}$(*<u>some man</u>* , < *smokes*, ∅ >)) =
F_{Concat}(*k*(*<u>some man</u>*), *k*(< *smokes*, ∅ >)) = (by (8))
F_{Concat}(($\lambda P_0\ \exists x_0$ ((**man'** x_0) & ($P_0\ x_0$))) , **smokes'**) =
(($\lambda P_0\ \exists x_0$ ((**man'** x_0) & ($P_0\ x_0$))) **smokes'**) ⇔ (by lambda-conversion)
$\exists x_0$ ((**man'** x_0) & (**smokes'** x_0))

(15) **Summary**
Well, it looks like we're all done here! We just need to properly implement the ideas in (5)-(12), and we can call it a day, put on our PJs, and get caught up on Breaking Bad!...

2. Towards a New Fragment of English: Quantificational NPs in Object Position

Oh wait… Crap! We forgot about the fact that quantificational terms can also be direct objects…

(16) **New Goal**
We would like for the sentences below to paired with the following translations:

a. <u>Sentence:</u> *<u>Michelle loves every president.</u>*
<u>Translation:</u> $\forall x_0$ ((**president'** x_0) → ((**loves'** x_0) **michelle'**))

b. <u>Sentence:</u> *<u>Mitt loves some man.</u>*
Translation: $\exists x_0$ ((**man'** x_0) & ((**loves'** x_0) **mitt'**))

(17) **The Obvious Problem**
In our translation base mapping Mini-English to Politics-NoQ, the syntactic operation $K_{\text{Merge-IV}}$ corresponds to the polynomial operation F_{Concat}.

- Given the translations in (8)-(10), if we maintain that correspondence, we'll end up mapping meaningful expressions of Mini-English to syntactic garbage in Politics+λ.
    - o And, we definitely don't get translations like (16a,b)

k(<u>*loves every president*</u>) =

k($K_{\text{Merge-IV}}$(< *loves*, ∅ > , <u>*every president*</u>)) =

F_{Concat}(*k*(< *loves*, ∅ >) , *k*(<u>*every president*</u>)) = (by (8))

F_{Concat}(**loves'** , ($\lambda P_0\ \forall x_0$ ((**president'** x_0) → ($P_0\ x_0$)))) =

(**loves'** ($\lambda P_0\ \forall x_0$ ((**president'** x_0) → ($P_0\ x_0$)))) = **syntactic garbage!**

(18) **The More General Issue: Quantificational Terms in Object Position**

- In the semantics for English that we get form our translation, we want terms like <u>*every president*</u> to end up denoting functions of type <<e,t>,t> (GQs)
- But we also in our semantics want transitive verbs like <u>*smokes*</u> to end up denoting functions of type <e,<e,t>> (curried characteristic functions of binary relations).
- **But, how do we put together meanings of those two different types?**

(19) **The PTQ Solution, Rough Idea: Part 1**

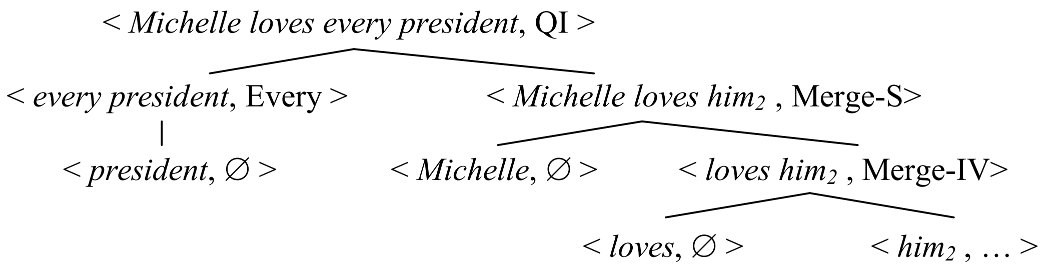
Our fragment of English will contain a syntactic operation that is like 'QR-in-reverse'.

- This operation K_{QI} (QI = 'quantifying in') will take (i) a quantificational term, and (ii) a sentence containing a pronoun (*cf.* trace), and output a string where the pronoun (*cf.* trace) is **replaced** with the quantification term.

<u>Rough Idea Illustrated:</u>

K_{QI}(<u>*every president*</u> , <u>*Michelle loves him*$_2$</u>) = <u>*Michelle loves every president.*</u>

(20) **Remark**
Although the transformation of the strings is like 'QR-in-reverse', the actual analysis trees output by K_{QI} are geometrically similar to the PS-trees generated by QR.


< *Michelle loves every president*, QI >
< *every president*, Every >
< *Michelle loves him₂* , Merge-S>
< *president*, ∅ >
< *Michelle*, ∅ >
< *loves him₂* , Merge-IV>
< *loves*, ∅ >
< *him₂* , … >

(21) **The PTQ Solution, Rough Idea: Part 2**
The polynomial operation corresponding to K_{QI} (H_{QI}) will do all the following:

(i) Take the translation of the quantificational term
(ii) Take the translation of the sentence
(iii) Lambda-abstract over the variable in the translation of the sentence
(iv) Apply the translation of the quantificational term to the resulting λ-expression

$k\,(K_{QI}(\,\underline{\textit{every president}}\,,\,\underline{\textit{Michelle loves him}_2}\,)\,)$ =

$H_{QI}\,(\,k(\underline{\textit{every president}})\,,\,k(\underline{\textit{Michelle loves him}_2})\,)$ =

The Key Step!

$H_{QI}\,((\lambda P_0\,\forall x_0\,((\textbf{president'}\,x_0) \rightarrow (P_0\,x_0)))\,,\,((\textbf{loves'}\,x_0)\,\textbf{michelle'})\,)$ =

$(\,(\lambda P_0\,\forall x_0\,((\textbf{president'}\,x_0) \rightarrow (P_0\,x_0)))\,(\lambda x_0\,((\textbf{loves'}\,x_0)\,\textbf{michelle'}))\,)$ ⇔
(by alpha-conversion)

$(\,(\lambda P_0\,\forall x_0\,((\textbf{president'}\,x_0) \rightarrow (P_0\,x_0)))\,(\lambda x_1((\textbf{loves'}\,x_1)\,\textbf{michelle'}))\,)$ ⇔
(by lambda-conversion)

$\forall x_0\,((\textbf{president'}\,x_0) \rightarrow ((\lambda x_1((\textbf{loves'}\,x_1)\,\textbf{michelle'}))\,x_0)))$ ⇔
(by lambda-conversion)

$\forall x_0\,((\textbf{president'}\,x_0) \rightarrow ((\textbf{loves'}\,x_0)\,\textbf{michelle'}))$

If we can flesh out the ideas in (19) and (21), we will have a system that properly translates / interprets quantificational terms in object position.

(22) **Our To-Do List**

a. Step One: Develop a theory of the syntax and semantics of pronouns in English
b. Step Two: Properly define the operation K_{QI}
c. Step Three: Properly define the operation H_{QI}

(23) **Fun Fact!**
Once we've set up everything correctly, our system will correctly predict quantifier-scope ambiguities. That is, our system will predict that the English sentence in (a) can receive either of the translations in (b).

a. Sentence of (Ambiguous) English: *Some man loves every president.*

b. Possible Translations into Politics+λ

(i) $\exists x_0\ ((\mathbf{man'}\ x_0)\ \&\ \forall x_1\ ((\mathbf{president'}\ x_1) \rightarrow ((\mathbf{loves'}\ x_1)\ x_0)))$

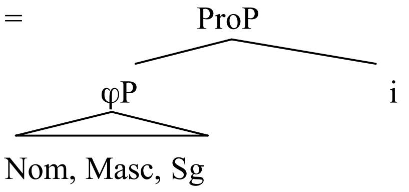
(ii) $\forall x_0\ ((\mathbf{president'}\ x_0) \rightarrow \exists x_1\ ((\mathbf{man'}\ x_1)\ \&\ ((\mathbf{loves'}\ x_0)\ x_1)))$

3. Developing The Analysis of English Quantification

3.1 The Syntax and Translation of English Pronouns

(24) **A Radical, Crazy Idea Only *Chomskians* Could Come Up With**
Perhaps pronouns are structurally complex, and consist of (i) a 'pronominal core' containing so-called 'φ-features' like Case, Gender, Number, and (ii) a pronominal index.

He$_i$ =



(25) **Montagovian Precursor (in "Universal Grammar")**
Pronouns in English are actually syntactically derived from more primitive 'indices'.

$K_{He}(i)$ = $\underline{He_i}$

(26) **Adapting This Idea to Our System: The Syntax of Indices**
We will assume that indices are trivial trees consisting of the following root-nodes:

$$\text{IND} = \{ < n , \varnothing > : n \in \mathbb{N} \}$$

(27) **The Translation of Pronouns into Politics+λ**
The system we will go on to build will not translate the pronouns *per se*, but rather the indices the pronouns are derived from.

a. <u>Translation of Indices:</u> For all $n \in \mathbb{N}$, $k(< n , \varnothing >) = x_n \quad (= v_{e,n})$

b. <u>Polynomial Operation for K_{He}:</u> $H_{He} = Id_{1,1}$

c. <u>Illustration:</u>
$k($ <u>*he 3*</u> $) = k (K_{He}(<3, \varnothing>)) =$
$H_{He}(k(<3, \varnothing>)) =$ (by (27a), (27b))
$Id_{1,1}(x_3) = x_3$

(28) **Partial Implementation of the Ideas in (25)-(27)**

a. <u>Expanding Our Set of Category Labels for English:</u>

(i)	TV	'transitive verb'
(ii)	IV	'intransitive verb' (and 'VPs')
(iii)	S	'sentence'
(iv)	T	'term' (formerly 'NP')
(v)	CN	'common noun' (that is, the 'Ns')
(vi)	**IN**	**'indices'**
(vii)	**PR**	**'pronouns'**

b. <u>The Basic Expressions:</u>

(i)	X_{TV}	=	{ < *loves*, ∅ > }
(ii)	X_{IV}	=	{ < *smokes*, ∅ > }
(iii)	X_{T}	=	{ < *Barack*, ∅ >, < *Mitt*, ∅ >, < *Michelle*, ∅ > }
(iv)	X_{CN}	=	{ < *man*, ∅ >, < *president*, ∅ > }
(v)	$\mathbf{X_{IN}}$	**=**	$\{ < \boldsymbol{n} , \varnothing > : \boldsymbol{n} \in \mathbb{N} \}$
(vi)	$\mathbf{X_{PR}}$	**=**	**∅**
(vii)	X_{S}	=	∅

c. <u>The Syntactic Operations:</u>
In addition to the operations we already have, let's add the operations K_{He} and K_{She}. We'll also revise the operation $K_{Merge\text{-}IV}$ as in (iii) below:

▪ In the definitions below, α and β are trees whose root nodes are ordered pairs. In addition α' and β' are the first members of the root nodes of α and β.

(i) $K_{He}(\alpha) =$
$$\begin{array}{c} < \textit{he}\ \alpha' , \text{He} > \\ | \\ \alpha \end{array}$$

(ii) $K_{She}(\alpha) =$
$$\begin{array}{c} < \textit{she}\ \alpha' , \text{She} > \\ | \\ \alpha \end{array}$$

(iii) $K_{\text{Merge-IV}}(\alpha, \beta)$ =

1. If β' = *he n*, then < α' *him n* , Merge-IV >

   α β

2. If β' = *she n*, then < α' *her n* , Merge-IV >

   α β

3. Otherwise < α' β' , Merge-IV >

   α β

<u>Note:</u> To capture the effects of pronominal case, we now have independent, empirical motivation for distinguishing the operations $K_{\text{Merge-IV}}$ and $K_{\text{Merge-S}}$

d. <u>The Syntactic Rules:</u>
Let us remove from our fragment of English the rules <$K_{\text{Merge-IV}}$, < TV, NP >, IV> and < $K_{\text{Merge-S}}$, < NP, IV >, S >. Let us add the following rules:

(i) < K_{He} , < IN >, PR >
‘Applying K_{He} to an expression of category IN (an index) yields an expression of category PR (a pronoun).’

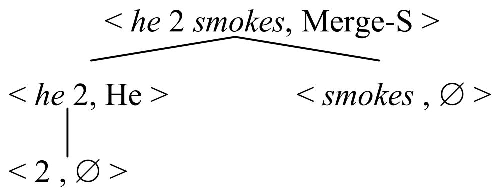
(ii) < K_{She} , < IN >, PR >
‘Applying K_{She} to an expression of category IN (an index) yields an expression of category PR (a pronoun).’

(iii) < $K_{\text{Merge-IV}}$, < TV, PR >, IV >
‘Applying $K_{\text{Merge-IV}}$ to an expression of category TV (transitive verb) and one of category PR (pronoun) yields an expression of category IV.’

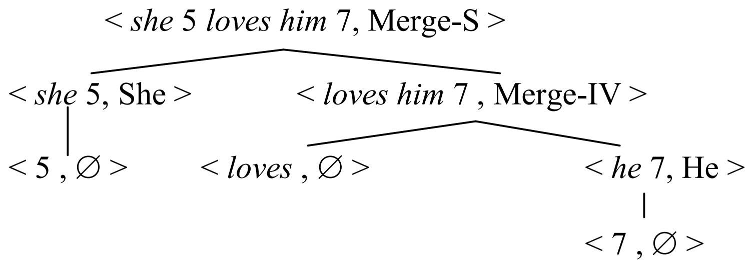
(iv) < $K_{\text{Merge-S}}$, < PR, IV >, S >
‘Applying $K_{\text{Merge-S}}$ to an expression of category PR (pronoun) and one of category IV yields an expression of category S.’

(29) **Illustration of the Resulting System**
The system sketched in (28) will entail that C_S contains trees such as:

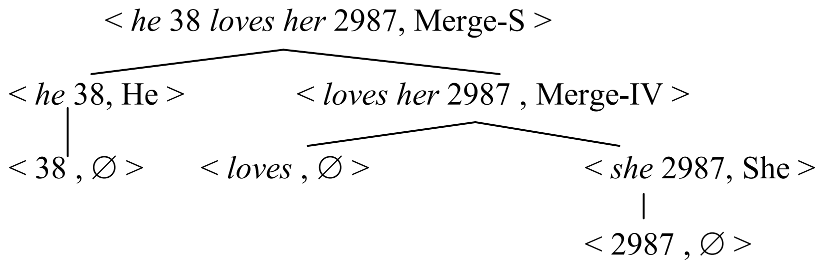
a.



b.

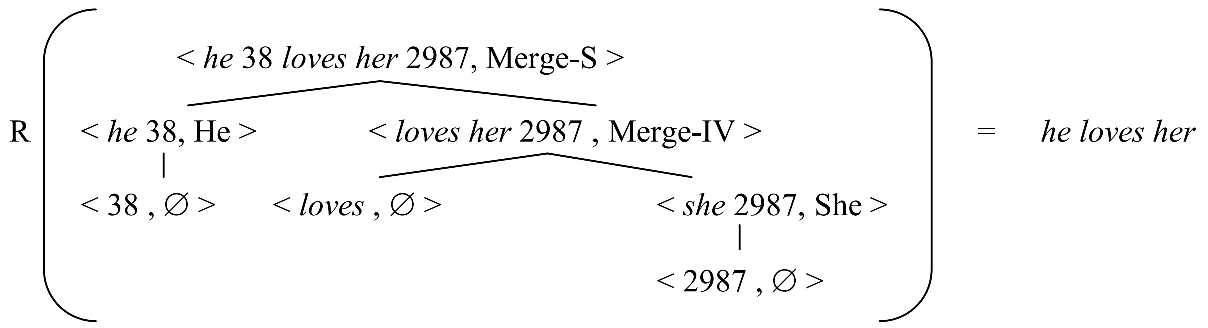


c.



Since we don't actually pronounce indices in English, we will want our 'ambiguating' relation R to delete them from the first member of the root-node of our trees.

(30) **A Slight Change to Our 'Ambiguating' Relation R**
R is a function which takes as input a tree T, and returns as output the first member of the root node of T *without any numerals*.



(31) **A Sketch of the Translation to Politics+λ**
The correspondences in (31a), added with (27a), will yield the translations in (31b).

a. Correspondences Between Syntactic Operations and Polynomial Operations:

(i)	K_{He}	~	H_{He}	=	$\text{Id}_{1,1}$
(ii)	K_{She}	~	H_{She}	=	$\text{Id}_{1,1}$
(iii)	$K_{\text{Merge-IV}}$	~	$H_{\text{Merge-IV}}$	=	F_{Concat}
(iv)	$K_{\text{Merge-S}}$	~	$H_{\text{Merge-S}}$	=	$F_{\text{Concat}}\langle\text{Id}_{2,2}, \text{Id}_{1,1}\rangle$

b. Illustrative Translations

(i) $k(\ \underline{\textit{he } 2 \textit{ smokes}}\ )$ =

$k(\ K_{\text{Merge-S}}(\ K_{\text{He}}(\langle 2, \varnothing\rangle), \langle \textit{smokes}, \varnothing \rangle\ )\ )$ =

$H_{\text{Merge-S}}(\ H_{\text{He}}(\ k(\langle 2, \varnothing\rangle)\ ), k(\langle \textit{smokes}, \varnothing \rangle)\ )$ =

$H_{\text{Merge-S}}(\ H_{\text{He}}(\ x_2\ ), \textbf{smokes'}\ )$ =

$F_{\text{Concat}}\langle\text{Id}_{2,2}, \text{Id}_{1,1}\rangle(\ \text{Id}_{1,1}(\ x_2\ ), \textbf{smokes'}\ )$ =

$F_{\text{Concat}}\langle\text{Id}_{2,2}, \text{Id}_{1,1}\rangle(\ x_2, \textbf{smokes'}\ )$ =

$(\ \textbf{smokes'}\ x_2\ )$

(ii) $k(\ \underline{\textit{she } 5 \textit{ loves him } 7}\ )$ =

$k(\ K_{\text{Merge-S}}(\ K_{\text{She}}(\langle 5, \varnothing\rangle),$
$K_{\text{Merge-IV}}(\langle \textit{loves}, \varnothing \rangle, K_{\text{He}}(\langle 7, \varnothing\rangle))))$ =

$H_{\text{Merge-S}}(\ H_{\text{She}}(\ k(\langle 5, \varnothing\rangle)\ ),$
$H_{\text{Merge-IV}}(\ k(\langle \textit{loves}, \varnothing \rangle), H_{\text{He}}(\ k(\langle 7, \varnothing\rangle)\ )))$ =

$H_{\text{Merge-S}}(\ H_{\text{She}}(\ x_5\ ),\ H_{\text{Merge-IV}}(\ \textbf{loves'}, H_{\text{He}}(\ x_7\ )))$ =

$F_{\text{Concat}}\langle\text{Id}_{2,2}, \text{Id}_{1,1}\rangle(\ \text{Id}_{1,1}(x_5), F_{\text{Concat}}(\ \textbf{loves'}, \text{Id}_{1,1}(x_7)))$ =

$F_{\text{Concat}}\langle\text{Id}_{2,2}, \text{Id}_{1,1}\rangle(\ x_5, F_{\text{Concat}}(\ \textbf{loves'}, x_7\ )))$ =

$((\textbf{loves'}\ x_7)\ x_5)$

3.2 The Syntactic Operation K_{QI} (‘Quantifying In’)

(32) **The Rough Idea Behind K_{QI}**
This will take (i) a quantificational term, and (ii) a sentence containing a pronoun, and output a string where the pronoun is **replaced** with the quantification term.

K_{QI}(<u>*every president*</u> , <u>*Michelle loves him$_2$*</u>) = <u>*Michelle loves every president.*</u>

(33) **Immediate Issue**
To properly define such an operation, there must be some means of specifying *which* pronoun in the sentential argument is to be replaced. *Here, there are two possible paths:*

a. <u>First Implementation (“Universal Grammar”):</u>
K_{QI} is a *ternary* relation, whose first argument is the index of the pronoun to be replaced in the sentential argument.

b. <u>Second Implementation (“PTQ”):</u>
Instead of a single operation K_{QI}, we have an *infinite* number of operations $K_{QI,n}$, where *n* is the index of the pronoun to be replaced in the sentential argument.

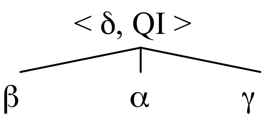
- In this unit, we’ll follow Montague in UG, and pursue the path in (33a).[1]

(36) **Preliminary Notation:** *Pro* = a meta-variable ranging over *he, she, him, her*

(37) **The Operation K_{QI}**

Let α, β, γ be trees whose root nodes are ordered pairs. In addition, let α’ , β’ , γ’ be the first members of the root nodes of α, β, γ (respectively).

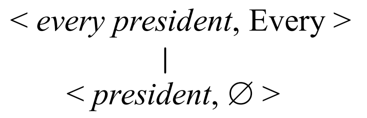
K_{QI} (α, β, γ) =



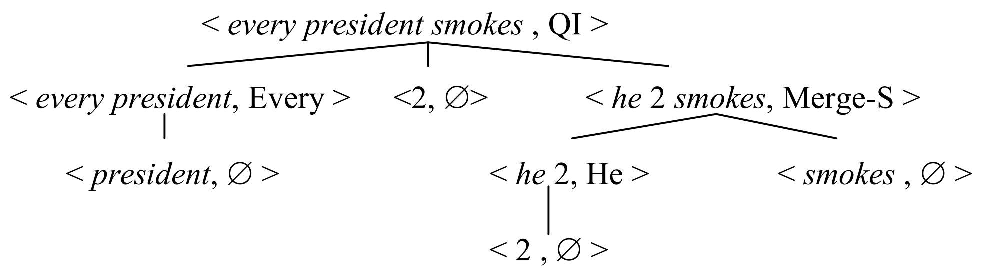
Where δ is the result of replacing in γ’ the left-most instance of *Pro* α’ with β’.

[1] The statement here is somewhat misleading, since Montague didn’t introduce a ‘quantifying-in’ operation in UG. However, the analysis Montague pursues for relative clauses in UG and PTQ faces the same issue in (33). The general approach in (33a) was the one taken up by Montague in UG for the operation forming relative clauses. The approach in (33b) was taken up in PTQ for both ‘quantifying-in’ and relativization.

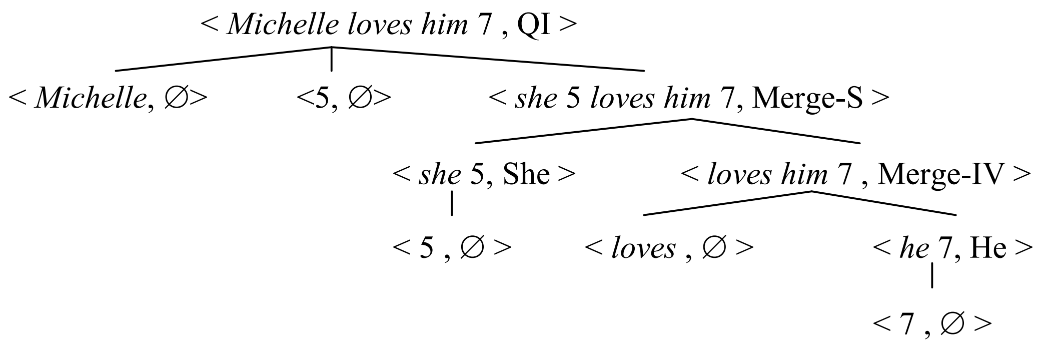
(38) **Illustration of K_{QI}**
In the examples below, let T_1 be the tree in (29a), T_2 be the tree in (29b). Finally, let T_3 be the tree:



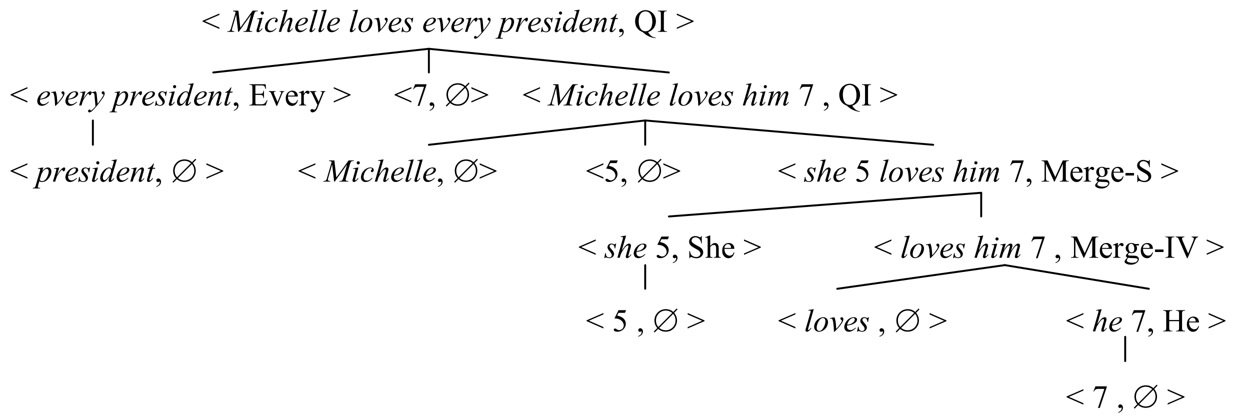
a. K_{QI}(<2, ∅> , T_3 , T_1) =


< *every president smokes* , QI >
< *every president*, Every >
<2, ∅>
< *he* 2 *smokes*, Merge-S >
< *president*, ∅ >
< *he* 2, He >
< *smokes* , ∅ >
< 2 , ∅ >

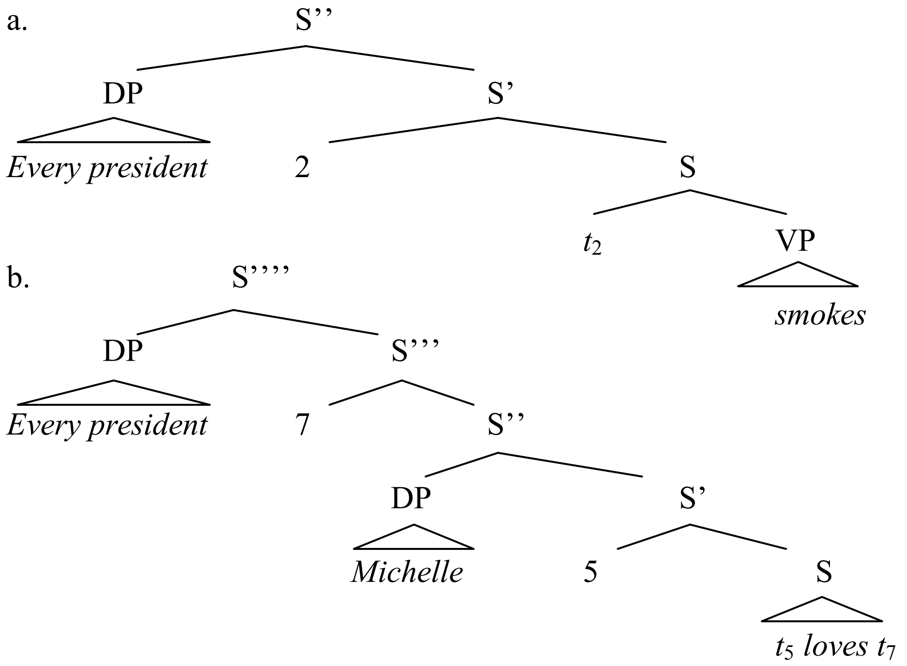
b. K_{QI}(<5, ∅> , < *Michelle*, ∅> , T_2) =


< *Michelle loves him* 7 , QI >
< *Michelle*, ∅>
<5, ∅>
< *she* 5 *loves him* 7, Merge-S >
< *she* 5, She >
< *loves him* 7 , Merge-IV >
< 5 , ∅ >
< *loves* , ∅ >
< *he* 7, He >
< 7 , ∅ >

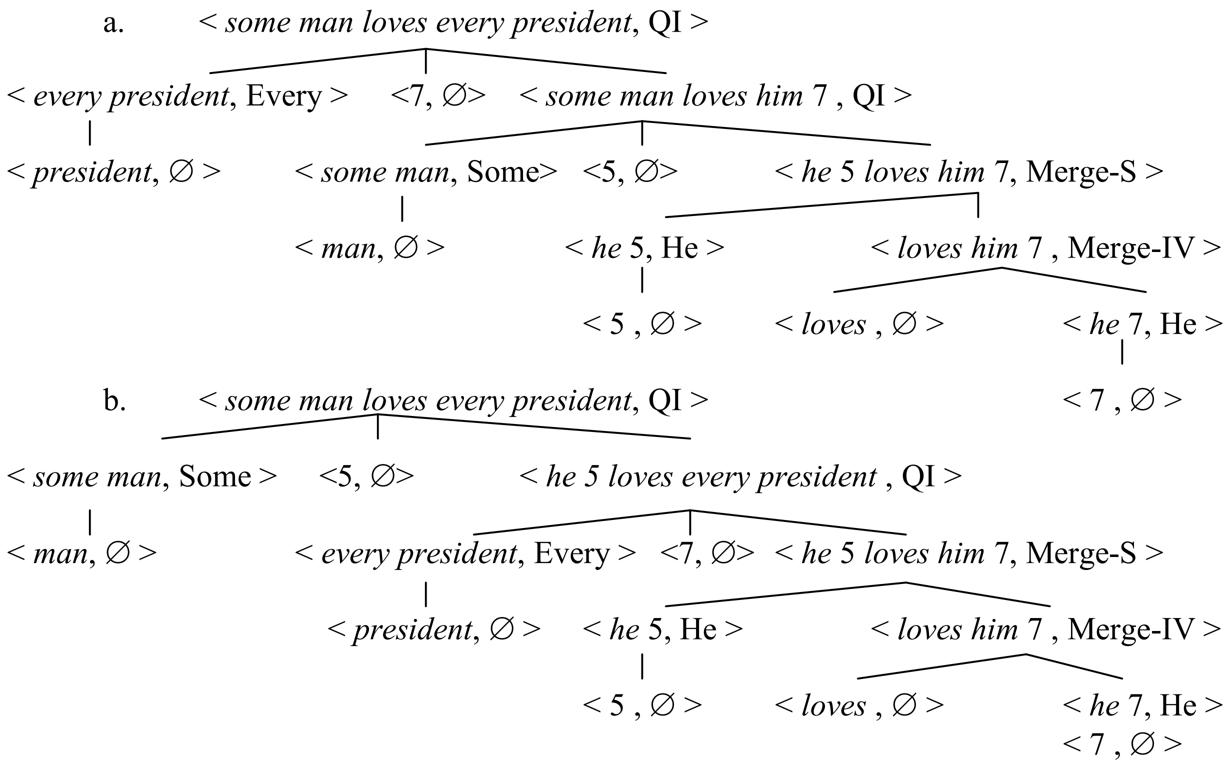
c. K_{QI}(<7, ∅> , T_3 , (38b)) =


< *Michelle loves every president*, QI >
< *every president*, Every >
<7, ∅>
< *Michelle loves him* 7 , QI >
< *president*, ∅ >
< *Michelle*, ∅>
<5, ∅>
< *she* 5 *loves him* 7, Merge-S >
< *she* 5, She >
< *loves him* 7 , Merge-IV >
< 5 , ∅ >
< *loves* , ∅ >
< *he* 7, He >
< 7 , ∅ >

(39) **Remark**
Notice how the analysis tree structures in (38) are geometrically similar to the LFs below, which would be posited in a 'Heim & Kratzer'-style system! (Hmmmm….)



(40) **The Roots of Quantifier Scope Ambiguities**
Our operation K_{QI} will be able to output *both* the following analysis tree structures. Note that the relation R in (30) would map both to the string *some man loves every president.*


a. < some man loves every president, QI >
< every president, Every >
< president, ∅ >
<7, ∅>
< some man loves him 7 , QI >
< some man, Some>
< man, ∅ >
<5, ∅>
< he 5 loves him 7, Merge-S >
< he 5, He >
< 5 , ∅ >
< loves him 7 , Merge-IV >
< loves , ∅ >
< he 7, He >
< 7 , ∅ >
b. < some man loves every president, QI >
< some man, Some >
< man, ∅ >
<5, ∅>
< he 5 loves every president , QI >
< every president, Every >
< president, ∅ >
<7, ∅>
< he 5 loves him 7, Merge-S >
< he 5, He >
< 5 , ∅ >
< loves him 7 , Merge-IV >
< loves , ∅ >
< he 7, He >
< 7 , ∅ >

(41) **Key Observation**
In the system we will ultimately develop from this, the English string *some man loves every president* is syntactically ambiguous.

- One analysis tree that R in (30) maps to this string is derived by QI-ing <u>*some man*</u> before <u>*every president*</u> (40a).

- Another analysis tree that R in (30) maps to this string is derived by QI-ing <u>*every president*</u> before <u>*some man*</u> (40b).

(42) **Key Idea**
We should design the polynomial operation H_{QI} so that this syntactic ambiguity will lead to a semantic ambiguity.

k ((40a)) = $\forall x_0$ ((**president'** x_0) → $\exists x_1$ ((**man'** x_1) & ((**loves'** x_0) x_1)))
k ((40b)) = $\exists x_0$ ((**man'** x_0) & $\forall x_1$ ((**president'** x_1) → ((**loves'** x_1) x_0)))

To close out this section, let us introduce the syntactic rule below.

(43) **The Syntactic Rule Employing K_{QI}**

< K_{QI} , < IN, T, S >, S >
'Applying K_{QI} to an expression of category IN (index), an expression of category T (term), and an expression of category S (sentence), yields and expression of category S.'

<u>Note:</u> Given this rule, the following are members of C_S: (38a), (38b), (38c), (40a), (40b)

3.3 The Polynomial Operation H_{QI} , Corresponding to K_{QI}

(44) **What We H_{QI} to Do For Us**
In general, if H_{QI}(x_n , φ , ψ) = (φ (λx_n ψ)), that would work well for our translation.

k(<u>*every president smokes*</u>) = (by (38a))

k (K_{QI}(<2, ∅> , <u>*every president*</u> , <u>*he 2 smokes*</u>)) = (by homomorphism property)

H_{QI}(k(<2, ∅>) , k(<u>*every president*</u>) , k(<u>*he 2 smokes*</u>))) = (by (27a), (8), (31b))

Key Part!

H_{QI}(x_2 , (λP_0 $\forall x_0$ ((**president'** x_0) → (P_0 x_0))) , (**smokes'** x_2)) = (by assumption)

((λP_0 $\forall x_0$ ((**president'** x_0) → (P_0 x_0))) (λx_2 (**smokes'** x_2))) ⇔ (by λ-conversion)

$\forall x_0$ ((**president'** x_0) → (**smokes'** x_0))

(45) **A Polynomial Operation that Would Do the Job**
Let us define H_{QI} as the following polynomial operation: $F_{Concat}< Id_{2,3} , F_{\lambda}< Id_{1,3}, Id_{3,3} > >$

Illustration:
$F_{Concat}<Id_{2,3}, F_{\lambda}<Id_{1,3}, Id_{3,3}>>(x_2 , (\lambda P_0\ \forall x_0\ ((\textbf{president'}\ x_0) \rightarrow (P_0\ x_0))) , (\textbf{smokes'}\ x_2)) =$

$F_{Concat} (Id_{2,3}(x_2 , (\lambda P_0\ \forall x_0\ ((\textbf{president'}\ x_0) \rightarrow (P_0\ x_0))) , (\textbf{smokes'}\ x_2)) ,$
$F_{\lambda}(\quad Id_{1,3}(x_2 , (\lambda P_0\ \forall x_0\ ((\textbf{president'}\ x_0) \rightarrow (P_0\ x_0))) , (\textbf{smokes'}\ x_2))$
$Id_{3,3}(x_2 , (\lambda P_0\ \forall x_0\ ((\textbf{president'}\ x_0) \rightarrow (P_0\ x_0))) , (\textbf{smokes'}\ x_2)))) \quad =$

$F_{Concat} ((\lambda P_0\ \forall x_0\ ((\textbf{president'}\ x_0) \rightarrow (P_0\ x_0))) , F_{\lambda}(x_2 , (\textbf{smokes'}\ x_2))) \quad =$

$F_{Concat} ((\lambda P_0\ \forall x_0\ ((\textbf{president'}\ x_0) \rightarrow (P_0\ x_0))) , (\lambda x_2\ (\textbf{smokes'}\ x_2))) \quad =$

$((\lambda P_0\ \forall x_0\ ((\textbf{president'}\ x_0) \rightarrow (P_0\ x_0))) (\lambda x_2\ (\textbf{smokes'}\ x_2)))$

We're almost ready to set up our fragment of English and the translation base. The last ingredient is to find polynomial operations corresponding to K_{Some} and K_{Every}

(46) **The Polynomial Operation H_{Some} , Corresponding to K_{Some}**

a. What We Want H_{Some} to Do For Us
In general, if $H_{Some}(\varphi) = (\lambda P_0\ \exists x_0\ ((\varphi\ x_0)\ \&\ (P_0\ x_0)))$, that would work well.

k(*some man*) = *k* (K_{Some}(<*man*, ∅>)) = H_{Some}(*k*(<*man*, ∅>)) = H_{Some}(**man'**) =
$(\lambda P_0\ \exists x_0\ ((\textbf{man'}\ x_0)\ \&\ (P_0\ x_0)))$

b. A Polynomial Operation that Would Do the Job
Let us define H_{Some} as the following polynomial operation:

$F_{\lambda}< C_{P0,1} , F_{\exists} < C_{x0,1} , F_{And} < F_{Concat}< Id_{1,1}, C_{x0,1}>, F_{Concat}< C_{P0,1} , C_{x0,1}>>>>$

Illustration:
$F_{\lambda}< C_{P0,1} , F_{\exists} < C_{x0,1} , F_{And} < F_{Concat}< Id_{1,1}, C_{x0,1}>, F_{Concat}< C_{P0,1} , C_{x0,1}>>>>(\textbf{man'}) \quad =$
$F_{\lambda}(C_{P0,1}(\textbf{man'}) , F_{\exists} (C_{x0,1}(\textbf{man'}) ,$
$F_{And}(\quad F_{Concat} (Id_{1,1}(\textbf{man'}), C_{x0,1}(\textbf{man'})) ,$
$F_{Concat} (C_{P0,1}(\textbf{man'}), C_{x0,1}(\textbf{man'}))))) \quad =$
$F_{\lambda}(P_0 , F_{\exists}(x_0 , F_{And}(F_{Concat}(\textbf{man'}, x_0) , F_{Concat} (P_0 , x_0)))) \quad =$
$F_{\lambda}(P_0 , F_{\exists}(x_0 , F_{And}((\textbf{man'}\ x_0) , (P_0\ x_0)))) \quad =$
$F_{\lambda}(P_0 , F_{\exists}(x_0 , ((\textbf{man'}\ x_0)\ \&\ (P_0\ x_0)))) \quad =$
$(\lambda P_0\ \exists x_0\ ((\textbf{man'}\ x_0)\ \&\ (P_0\ x_0)))$

(47) **The Polynomial Operation H_{Every} , Corresponding to K_{Every}**

a. What We Want H_{Every} to Do For Us
In general, if $H_{\text{Every}}(\varphi) = (\lambda P_0\ \forall x_0\ ((\varphi\ x_0) \rightarrow (P_0\ x_0)))$, that would work well.

k(*every man*) = k (K_{Every} (<*man*, ∅>)) = H_{Every}(k(<*man*, ∅>)) = H_{Every}(**man'**) = ($\lambda P_0\ \forall x_0$ ((**man'** x_0) $\rightarrow$ ($P_0\ x_0$)))

b. A Polynomial Operation that Would Do the Job
Let us define H_{Every} as the following polynomial operation:

$F_\lambda< C_{P0,1}\ , F_\forall < C_{x0,1}\ , F_{\text{If}} < F_{\text{Concat}} < Id_{1,1}, C_{x0,1}>, F_{\text{Concat}} < C_{P0,1}\ , C_{x0,1}>>>>$

Illustration:

$F_\lambda< C_{P0,1}\ , F_\forall < C_{x0,1}\ , F_{\text{If}} < F_{\text{Concat}} < Id_{1,1}, C_{x0,1}>, F_{\text{Concat}} < C_{P0,1}\ , C_{x0,1}>>>>$(**man'**) =
F_λ($C_{P0,1}$(**man'**) , $F_\forall$ ($C_{x0,1}$(**man'**) ,
F_{If}(F_{Concat} ($Id_{1,1}$(**man'**), $C_{x0,1}$(**man'**)) ,
F_{Concat} ($C_{P0,1}$(**man'**), $C_{x0,1}$(**man'**)))))) =
F_λ(P_0 , $F_\forall$ (x_0 , F_{If}(F_{Concat}(**man'**, x_0) , F_{Concat} (P_0 , x_0))))) =
F_λ(P_0 , $F_\forall$ (x_0 , F_{If}((**man'** x_0) , (P_0 x_0))))) =
F_λ(P_0 , $F_\forall$ (x_0 , ((**man'** x_0) $\rightarrow$ (P_0 x_0)))) =
($\lambda P_0\ \forall x_0$ ((**man'** x_0) $\rightarrow$ ($P_0\ x_0$)))

We're now ready to put all these ideas together into a semantic analysis of English quantification!

4. The Fragment of English: 'Mini-English+Qs' (ME+Q)

(48) **Step One: The Category Labels** Δ = {TV, IV, S, T, CN, IN, PR}

(49) **Step Two: The Basic Expressions**

X_{TV}	=	{ < *loves*, ∅ > }
X_{IV}	=	{ < *smokes*, ∅ > }
X_T	=	{ < *Barack*, ∅ >, < *Mitt*, ∅ >, < *Michelle*, ∅ > }
X_{CN}	=	{ < *man*, ∅ >, < *president*, ∅ > }
X_{IN}	=	{ < n , ∅ > : $n \in \mathbb{N}$ }
X_{PR}	=	∅
X_S	=	∅

(50) **Step Three: The Syntactic Operations**

a. Operations Unchanged From Disambiguated Mini-English (DME)

$K_{\text{Merge-S}}$ = (as defined previously)
K_{Not} = (as defined previously)
K_{And} = (as defined previously)
K_{If} = (as defined previously)

b. New Syntactic Operations

K_{Some} = (as defined in (5c))
K_{Every} = (as defined in (5c))
K_{He} = (as defined in (28c))
K_{She} = (as defined in (28c))
$K_{\text{Merge-IV}}$ = (as defined in (28c))
K_{QI} = (as defined in (37))

$K_{\text{Or}}(\alpha,\beta)$ =

$$\begin{tikzcd}[column sep=small] & < \alpha' \textit{ or } \beta', \text{Or} > \arrow[dl, dash] \arrow[dr, dash] & \\ \alpha & & \beta \end{tikzcd}$$

Another new one we'll add, just for fun!

(51) **Step Four: The Syntactic Algebra**

$< E, K_\gamma >_{\gamma \in \{\text{Merge-S, Merge-IV, Not, And, If, Or, Some, Every, He, She, QI}\}}$ is the algebra such that:

a. $\{ K_\gamma \}_{\gamma \in \{\text{Merge-S, Merge-IV, Not, And, If, Or, Some, Every, He, She, QI}\}}$ are as defined above, and

b. E is the smallest set such that
 (i) For all $\delta \in \Delta$, $X_\delta \subseteq E$
 (ii) E is closed under $\{ K_\gamma \}_{\gamma \in \{\text{Merge-S, Merge-IV, Not, And, If, Or, Some, Every, He, She, QI}\}}$

(52) **Step Five: The Syntactic Rules**

The set S_E consists of the following triples:

a. $< K_{\text{Merge-S}}, < \text{PR, IV} >, \text{S} >$
b. $< K_{\text{Merge-IV}}, < \text{TV, PR} >, \text{IV} >$
c. $< K_{\text{Not}}, < \text{S} >, \text{S} >$
d. $< K_{\text{And}}, < \text{S, S} >, \text{S} >$
e. $< K_{\text{If}}, < \text{S, S} >, \text{S} >$
f. $< K_{\text{Or}}, < \text{S, S} >, \text{S} >$
g. $< K_{\text{Some}} , < \text{CN} >, \text{T} >$
h. $< K_{\text{Every}} , < \text{CN} >, \text{T} >$
i. $< K_{\text{He}} , < \text{IN} >, \text{PR} >$
j. $< K_{\text{She}} , < \text{IN} >, \text{PR} >$
k. $< K_{\text{QI}} , < \text{IN, T, S} >, \text{S} >$

Note: Contrary to the picture in (12)-(14), the rules above do not allow us to $K_{\text{Merge-S}}$ a term T with an IV to form an S.

- We could easily add such a rule, but will refrain from doing so until later…

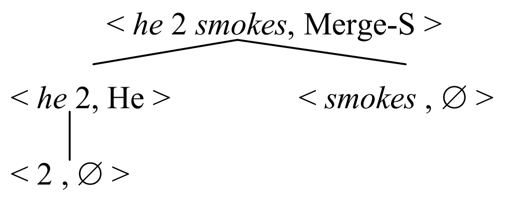
(53) **The Disambiguated Language: 'Disambiguated Mini-English+Qs' (DME+Q)**

The structure $< E, K_{\gamma}, X_{\delta}, S_E, S >_{\gamma \in \{\text{Merge-S, Merge-IV, Not, And, If, Or, Some, Every, He, She, QI}\}, \delta \in \Delta}$ where E, K_{γ}, X_{δ}, S_E, and Δ are as defined in (48)-(52).

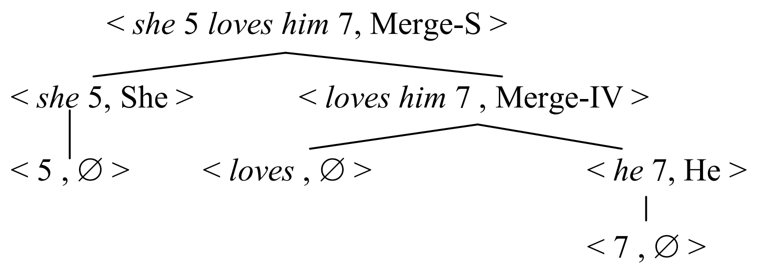
(54) **Some Illustrative Members of C_S**

Students are encouraged to work out for themselves how the rules in (52) entail that the following are all members of C_S in DME+Q.

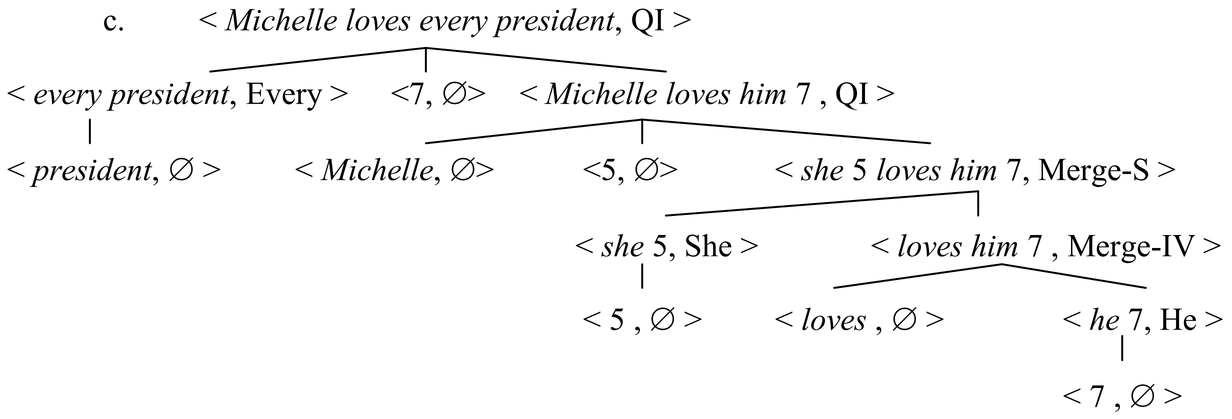
a.



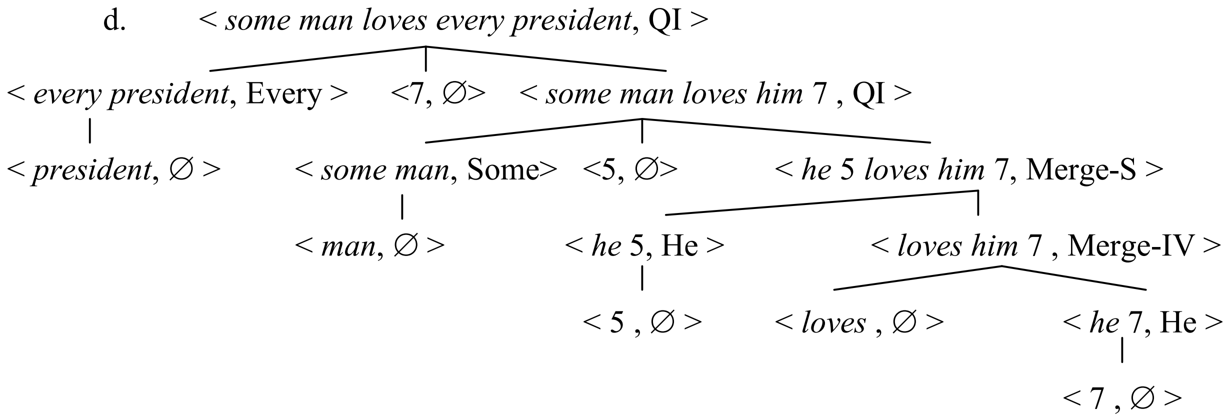
b.



c.


< *Michelle loves every president*, QI >
< *every president*, Every >
<7, ∅>
< *Michelle loves him* 7 , QI >
< *president*, ∅ >
< *Michelle*, ∅>
<5, ∅>
< *she* 5 *loves him* 7, Merge-S >
< *she* 5, She >
< *loves him* 7 , Merge-IV >
< 5 , ∅ >
< *loves* , ∅ >
< *he* 7, He >
< 7 , ∅ >

d.


< *some man loves every president*, QI >
< *every president*, Every >
<7, ∅>
< *some man loves him* 7 , QI >
< *president*, ∅ >
< *some man*, Some>
<5, ∅>
< *he* 5 *loves him* 7, Merge-S >
< *man*, ∅ >
< *he* 5, He >
< *loves him* 7 , Merge-IV >
< 5 , ∅ >
< *loves* , ∅ >
< *he* 7, He >
< 7 , ∅ >

(55) **The Fragment of English: Mini-English+Qs (ME+Q)**
Mini-English+Qs is the following language, where R is as defined in (30):
$<< E, K_\gamma, X_\delta, S_E, S >_{\gamma \in \{\text{Merge-S, Merge-IV, Not, And, If, Or, Some, Every, He, She, QI}\}, \delta \in \Delta}, R>$

(56) **Some Illustrative Members of C_S in ME+Q**
a. *he smokes*
b. *she loves him*
c. *Michelle loves every president*
d. *some man loves every president*

Note: As was observed in (41), sentence (56d) of ME+Q is syntactically ambiguous.

5. The Translation Base from Mini-English+Qs to Politics+λ

(57) **Step One: The Syntactic Category Mapping**
Let the function $g: \Delta \rightarrow T \cup \{ <var,\tau> : \tau \in T \}$ be defined as follows:

a. $g(TV) = <e, <e,t>>$
b. $g(IV) = <e, t>$
c. $g(S) = t$
d. $g(T) = < <e,t>, t >$
e. $g(CN) = <e, t>$
f. $g(IN) = <var, e>$
g. $g(PR) = e$

(58) **Step Two: The Polynomial Operations**
Let $\{ H_\gamma \}_{\gamma \in \{\text{Merge-S, Merge-IV, Not, And, If, Or, Some, Every, He, She, QI}\}}$ be the following polynomial operations over the syntactic algebra $<A, F_\gamma>_{\gamma \in \{\text{Concat, Not, And, Or, If}, \exists, \forall, \lambda\}}$ for Politics+λ.

a. $H_{\text{Merge-S}} = F_{\text{Concat}}<Id_{2,2}, Id_{1,1}>$
b. $H_{\text{Merge-IV}} = F_{\text{Concat}}$
c. $H_{\text{Not}} = F_{\text{Not}}$
d. $H_{\text{And}} = F_{\text{And}}$
e. $H_{\text{Or}} = F_{\text{Or}}$
f. $H_{\text{If}} = F_{\text{If}}$
g. $H_{\text{Some}} =$
$F_\lambda< C_{P0,1} , F_\exists < C_{x0,1} , F_{\text{And}} < F_{\text{Concat}}< Id_{1,1}, C_{x0,1}>, F_{\text{Concat}}< C_{P0,1} , C_{x0,1}>>>>$
h. $H_{\text{Every}} =$
$F_\lambda< C_{P0,1} , F_\forall < C_{x0,1} , F_{\text{If}} < F_{\text{Concat}}< Id_{1,1}, C_{x0,1}>, F_{\text{Concat}}< C_{P0,1} , C_{x0,1}>>>>$
i. $H_{\text{He}} = Id_{1,1}$
j. $H_{\text{She}} = Id_{1,1}$
k. $H_{\text{QI}} = F_{\text{Concat}}< Id_{2,3} , F_\lambda< Id_{1,3}, Id_{3,3} > >$

(59) **Step Three: Checking for Derived Syntactic Rules**
In order to employ the polynomial operations above in our translation base, it must the case that each of the following are derived syntactic rules of Politics+λ.
Students are encouraged to confirm for themselves whether they are.

a. $< F_{\text{Concat}} <\text{Id}_{2,2}, \text{Id}_{1,1}>, < e, <e,t> >, t >$ (~ Rule (52a))
b. $< F_{\text{Concat}} , < <e,<e,t>>, e >, <e,t> >$ (~ Rule (52b))
c. $< F_{\text{Not}} , < t >, t >$ (~ Rule (52c))
d. $< F_{\text{And}}, < t, t >, t >$ (~ Rule (52d))
e. $< F_{\text{If}}, < t, t >, t >$ (~ Rule (52e))
f. $< F_{\text{Or}}, < t, t >, t >$ (~ Rule (52f))
g. $< F_{\lambda}< C_{P0,1} , F_{\exists} < C_{x0,1} , F_{\text{And}} < F_{\text{Concat}}< \text{Id}_{1,1}, C_{x0,1}>, F_{\text{Concat}}< C_{P0,1} , C_{x0,1}>>>>,$
$< <e,t> >, <<e,t>, t> >$ (~ Rule (52g))
h. $< F_{\lambda}< C_{P0,1} , F_{\forall} < C_{x0,1} , F_{\text{If}} < F_{\text{Concat}}< \text{Id}_{1,1}, C_{x0,1}>, F_{\text{Concat}}< C_{P0,1} , C_{x0,1}>>>>,$
$< <e,t> >, <<e,t>, t> >$ (~ Rule (52h))
i. $< \text{Id}_{1,1} < <var,e> > e >$ (~ Rule (52i) and (52j)) [2]
j. $< F_{\text{Concat}}< \text{Id}_{2,3} , F_{\lambda}< \text{Id}_{1,3}, \text{Id}_{3,3}>> , < <var,e>, <<e,t>, t>, t>, t >$ (~ Rule (52k))

(60) **Step Four: The Lexical Translation Function**
The function j has as its domain $\{ X_{\delta} \}_{\delta \in \Delta}$, and consists of the following mappings:

a.	j(< *loves*, ∅ >)	=	**loves'**
b.	j(< *smokes*, ∅ >)	=	**smokes'**
c.	j(< *Barack*, ∅ >)	=	(λP_0 (P_0 **barack'**))
d.	j(< *Mitt*, ∅ >)	=	(λP_0 (P_0 **mitt'**))
e.	j(< *Michelle*, ∅ >)	=	(λP_0 (P_0 **michelle'**))
f.	j(< *man*, ∅ >)	=	**man'**
g.	j(< *president*, ∅ >)	=	**president'**
h.	For all $n \in \mathbb{N}$, j(< *n*, ∅ >)	=	x_n (= $v_{e,n}$)

(61) **Definition of the Translation Base from Mini-English+Qs to Politics+λ**
Let **T** be the structure $< g , H_{\gamma} , j>_{\gamma \in \{\text{Merge-S, Merge-IV, Not, And, If, Or, Some, Every, He, She, QI}\}}$, where g, H_{γ} , and j are as defined in (57)-(60). **T** is a translation base from Mini-English+Qs to Politics+λ.

We can now use the translation function k determined by **T** *to homomorphically map expressions of DME+Q (analysis trees) to expressions of Politics+λ.*

[2] There's actually a problem here for our definition of 'derived syntactic rule'. Although (59i) is true when informally read ('Applying $\text{Id}_{1,1}$ to an expression of category <var,e> yields an expression of category e'), it is not strictly speaking a 'derived rule of Politics+λ', since the *label* <var,e> ≠ e. **I don't myself know of any possible solutions to this problem.** (Montague himself doesn't run into it in UG, for various independent reasons…)

(62) **A Critical Inference Rule When Translating: Alpha-Conversion**
If variable *v* is bound in φ, and variable *v'* appears nowhere in φ, then φ is logically equivalent to $[v/v']\varphi$

Illustration: $(\lambda P_0\ \exists x_0\ ((\textbf{man'}\ x_0)\ \&\ (P_0\ x_0)))\ \Leftrightarrow\ (\lambda P_1\ \exists x_1\ ((\textbf{man'}\ x_1)\ \&\ (P_1\ x_1)))$

(63) **Illustration of the Translation Function *k***
Let T_1 be the tree for *some man loves every president* in (40a) [every > some].

(i) $k(T_1)$ =

(ii) $k(\ K_{QI}(<7, \emptyset>, K_{Every}(<\textit{president}, \emptyset>),$
$K_{QI}(<5, \emptyset>, K_{Some}(<\textit{man}, \emptyset>),$
$K_{Merge\text{-}S}(\ K_{He}(<5\ , \emptyset>), K_{Merge\text{-}IV}(<\textit{loves}\ , \emptyset>, K_{He}(<7\ , \emptyset>))))))$ =

(iii) $H_{QI}(\ k(<7, \emptyset>), H_{Every}(\ k(<\textit{president}, \emptyset>)\ ),$
$H_{QI}(\ k\ (<5, \emptyset>), H_{Some}(\ k(<\textit{man}, \emptyset>)\ ),$
$H_{Merge\text{-}S}(\ H_{He}(\ k(<5\ , \emptyset>)\ ), H_{Merge\text{-}IV}(\ k(<\textit{loves}\ , \emptyset>), H_{He}(\ k(<7\ , \emptyset>))))))$ =

(iv) $H_{QI}(\ j(<7, \emptyset>), H_{Every}(\ j(<\textit{president}, \emptyset>)\ ),$
$H_{QI}(\ j(<5, \emptyset>), H_{Some}(\ j(<\textit{man}, \emptyset>)\ ),$
$H_{Merge\text{-}S}(\ H_{He}(\ j(<5\ , \emptyset>)\ ), H_{Merge\text{-}IV}(\ j(<\textit{loves}\ , \emptyset>), H_{He}(\ j(<7\ , \emptyset>))))))$ =

(v) $H_{QI}(\ x_7\ , H_{Every}(\textbf{president'}),$
$H_{QI}(\ x_5\ , H_{Some}(\textbf{man'}),$
$H_{Merge\text{-}S}(\ H_{He}(x_5), H_{Merge\text{-}IV}(\ \textbf{loves'}\ , H_{He}(x_7)))))$ =

(vi) $H_{QI}(\ x_7\ , H_{Every}(\textbf{president'}),$
$H_{QI}(\ x_5\ , H_{Some}(\textbf{man'}), ((\textbf{loves'}\ x_7)\ x_5)\ )$ =

(vii) $H_{QI}(\ x_7\ , (\lambda P_0\ \forall x_0\ ((\textbf{president'}\ x_0) \rightarrow (P_0\ x_0))),$
$H_{QI}(\ x_5\ , (\lambda P_0\ \exists x_0\ ((\textbf{man'}\ x_0)\ \&\ (P_0\ x_0))), ((\textbf{loves'}\ x_7)\ x_5)\ )$ =

(viii) $H_{QI}(\ x_7\ , (\lambda P_0\ \forall x_0\ ((\textbf{president'}\ x_0) \rightarrow (P_0\ x_0))),$
$(\ (\lambda P_0\ \exists x_0\ ((\textbf{man'}\ x_0)\ \&\ (P_0\ x_0)))\ (\lambda x_5\ ((\textbf{loves'}\ x_7)\ x_5))\ )\ )$ =

(ix) $((\lambda P_0\ \forall x_0\ ((\textbf{president'}\ x_0) \rightarrow (P_0\ x_0)))$
$(\lambda x_7\ ((\lambda P_0\ \exists x_0\ ((\textbf{man'}\ x_0)\ \&\ (P_0\ x_0)))\ (\lambda x_5\ ((\textbf{loves'}\ x_7)\ x_5)))))$

(64) **Remark**
The end product of our translation process is the complex lambda-expression in (63ix). However, thanks to α-conversion and λ-conversion, we can 'transform' this into a simpler, logically equivalent formula.

Note:
Because of all the variables shared between the sub-formulae in (63ix), we cannot immediately apply λ-conversion to 'simplify' the formula; instead, we must first apply α-conversion to change the bound variables *so that they aren't shared between the lambda-expressions.*

(64) **Simplifying the Translation in (63)**

(i) $((\lambda P_0\ \forall x_0\ ((\textbf{president'}\ x_0) \rightarrow (P_0\ x_0)))$
$(\lambda x_7\ ((\lambda P_0\ \exists x_0\ ((\textbf{man'}\ x_0)\ \&\ (P_0\ x_0)))\ (\lambda x_5\ ((\textbf{loves'}\ x_7)\ x_5)))))$ ⇔ (α-conversion)

(ii) $((\lambda P_0\ \forall x_0\ ((\textbf{president'}\ x_0) \rightarrow (P_0\ x_0)))$
$(\lambda x_7\ ((\lambda P_1\ \exists x_1\ ((\textbf{man'}\ x_1)\ \&\ (P_1\ x_1)))\ (\lambda x_5\ ((\textbf{loves'}\ x_7)\ x_5)))))$ ⇔ (λ-conversion)

(iii) $\forall x_0\ ((\textbf{president'}\ x_0) \rightarrow$
$((\lambda x_7\ ((\lambda P_1\ \exists x_1\ ((\textbf{man'}\ x_1)\ \&\ (P_1\ x_1)))\ (\lambda x_5\ ((\textbf{loves'}\ x_7)\ x_5))))\ x_0))$ ⇔ (λ-conversion)

(iv) $\forall x_0\ ((\textbf{president'}\ x_0) \rightarrow$
$((\lambda P_1\ \exists x_1\ ((\textbf{man'}\ x_1)\ \&\ (P_1\ x_1)))(\lambda x_5\ ((\textbf{loves'}\ x_0)\ x_5))))$ ⇔ (λ-conversion)

(v) $\forall x_0\ ((\textbf{president'}\ x_0) \rightarrow$
$\exists x_1\ ((\textbf{man'}\ x_1)\ \&\ ((\lambda x_5\ ((\textbf{loves'}\ x_0)\ x_5))\ x_1)))$ ⇔ (λ-conversion)

(vi) $\forall x_0\ ((\textbf{president'}\ x_0) \rightarrow \exists x_1\ ((\textbf{man'}\ x_1)\ \&\ ((\textbf{loves'}\ x_0)\ x_1)))$

(65) **Remark**
- Again, strictly speaking, the *translation* of tree (40a) is the complex lambda expression in (63ix).
  - o The simpler formula in (64iv) is **not** the translation of (40a).
- (65vi) is however, a logically-equivalent formula, one that allows us to more easily 'see' what the semantic value induced by our translation for (40a) is…

(66) **Illustration of the Translation Function *k***
Let T_2 be the tree for *some man loves every president* in (40b) [some > every].

(i) $k(T_1)$ =

(ii) $k(\ K_{QI}(<5, \varnothing>, K_{Some}(<man, \varnothing>),$
$K_{QI}(<7, \varnothing>, K_{Every}(<president, \varnothing>),$
$K_{Merge\text{-}S}(\ K_{He}(<5\ , \varnothing>), K_{Merge\text{-}IV}(<loves\ , \varnothing>, K_{He}(<7\ , \varnothing>))))))$ =

(iii) $H_{QI}(\ k(<5, \varnothing>), H_{Some}(\ k(<man, \varnothing>)\ ),$
$H_{QI}(\ k\ (<7, \varnothing>), H_{Every}(\ k(<president, \varnothing>)\ ),$
$H_{Merge\text{-}S}(\ H_{He}(\ k(<5\ , \varnothing>)\ ), H_{Merge\text{-}IV}(\ k(<loves\ , \varnothing>), H_{He}(\ k(<7\ , \varnothing>))))))$ =

(iv) $H_{\text{QI}}(\, j(<5, \varnothing>), H_{\text{Some}}(\, j(<\textit{man}, \varnothing>)\,),$
$H_{\text{QI}}(\, j\,(<7, \varnothing>), H_{\text{Every}}(\, j(<\textit{president}, \varnothing>)\,),$
$H_{\text{Merge-S}}(\, H_{\text{He}}(\, j(<5\,, \varnothing>)\,), H_{\text{Merge-IV}}(\, j(<\textit{loves}\,, \varnothing>), H_{\text{He}}(\, j(<7\,, \varnothing>))))))$ =

(v) $H_{\text{QI}}(\, x_5\,, H_{\text{Some}}(\textbf{man'}),$
$H_{\text{QI}}(\, x_7\,, H_{\text{Every}}(\textbf{president'}),$
$H_{\text{Merge-S}}(\, H_{\text{He}}(x_5), H_{\text{Merge-IV}}(\, \textbf{loves'}\,, H_{\text{He}}(x_7)))))$ =

(vi) $H_{\text{QI}}(\, x_5\,, H_{\text{Some}}(\textbf{man'}),$
$H_{\text{QI}}(\, x_7\,, H_{\text{Every}}(\textbf{president'}), ((\textbf{loves'}\; x_7)\; x_5)\,)$ =

(vii) $H_{\text{QI}}(\, x_5\,, (\lambda P_0\; \exists x_0\; ((\textbf{man'}\; x_0)\; \&\; (P_0\, x_0))),$
$H_{\text{QI}}(\, x_7\,, (\lambda P_0\; \forall x_0\; ((\textbf{president'}\; x_0) \rightarrow (P_0\, x_0))), ((\textbf{loves'}\; x_7)\; x_5)\,)$ =

(viii) $H_{\text{QI}}(\, x_5\,, (\lambda P_0\; \exists x_0\; ((\textbf{man'}\; x_0)\; \&\; (P_0\, x_0))),$
$((\lambda P_0\; \forall x_0\; ((\textbf{president'}\; x_0) \rightarrow (P_0\, x_0)))\; (\lambda x_7\, ((\textbf{loves'}\; x_7)\; x_5))))$ =

(ix) $((\lambda P_0\; \exists x_0\; ((\textbf{man'}\; x_0)\; \&\; (P_0\, x_0)))$
$(\lambda x_5\, ((\lambda P_0\; \forall x_0\; ((\textbf{president'}\; x_0) \rightarrow (P_0\, x_0)))\; (\lambda x_7\, ((\textbf{loves'}\; x_7)\; x_5))))$ =

Once again, we can apply α-conversion and λ-conversion to 'transform' the translation in (ix) into a simpler, logically-equivalent formula

(x) $((\lambda P_0\; \exists x_0\; ((\textbf{man'}\; x_0)\; \&\; (P_0\, x_0)))$
$(\lambda x_5\, ((\lambda P_0\; \forall x_0\; ((\textbf{president'}\; x_0) \rightarrow (P_0\, x_0)))\; (\lambda x_7\, ((\textbf{loves'}\; x_7)\; x_5))))$ ⇔ (α-conversion)

(xi) $((\lambda P_0\; \exists x_0\; ((\textbf{man'}\; x_0)\; \&\; (P_0\, x_0)))$
$(\lambda x_5\, ((\lambda P_1\; \forall x_1\; ((\textbf{president'}\; x_1) \rightarrow (P_1\, x_1)))\; (\lambda x_7\, ((\textbf{loves'}\; x_7)\; x_5))))$ ⇔ (λ-conversion)

(xii) $\exists x_0\; ((\textbf{man'}\; x_0)\; \&$
$((\lambda x_5\, ((\lambda P_1\; \forall x_1\; ((\textbf{president'}\; x_1) \rightarrow (P_1\, x_1)))\; (\lambda x_7\, ((\textbf{loves'}\; x_7)\; x_5)))\; x_0))$ ⇔ (λ-conversion)

(xiii) $\exists x_0\; ((\textbf{man'}\; x_0)\; \&$
$((\lambda P_1\; \forall x_1\; ((\textbf{president'}\; x_1) \rightarrow (P_1\, x_1)))\; (\lambda x_7\, ((\textbf{loves'}\; x_7)\; x_0)))$ ⇔ (λ-conversion)

(xiv) $\exists x_0\; ((\textbf{man'}\; x_0)\; \&$
$\forall x_1\; ((\textbf{president'}\; x_1) \rightarrow ((\lambda x_7\, ((\textbf{loves'}\; x_7)\; x_0))\; x_1)))$ ⇔ (λ-conversion)

(xv) $\exists x_0\; ((\textbf{man'}\; x_0)\; \&\; \forall x_1\; ((\textbf{president'}\; x_1) \rightarrow ((\textbf{loves'}\; x_1)\; x_0)))$

Again, the formula in (63xiv) is not the *translation* of tree (40b); the translation is (63ix).
However* (63xv) *is a simpler, logically-equivalent expression…
***Thus, it offers a more 'transparent' representation of the meaning assigned to* (40b)**

6. Inducing an Interpretation for Mini-English+Q

If we compose together our translation function *k* and our meaning assignment *h* for Politics+λ, we now (thanks to our general theory of translation), obtain a meaning assignment for DME+Q.

(67) **Illustration of the Induced Interpretation of Mini-English+Q**

Let T_1 be the tree for *some man loves every president* in (40a) [every > some].

(i) $h^\circ k(T_1)$ = (by definition of composition)

(ii) $h(k(T_1))$ = (by (63))

(iii) $h(((\lambda P_0\ \forall x_0\ ((\textbf{president'}\ x_0) \rightarrow (P_0\ x_0)))$
$(\lambda x_7\ ((\lambda P_0\ \exists x_0\ ((\textbf{man'}\ x_0)\ \&\ (P_0\ x_0)))\ (\lambda x_5\ ((\textbf{loves'}\ x_7)\ x_5))))))$ = (by (64))

(iv) $h(\forall x_0\ ((\textbf{president'}\ x_0) \rightarrow \exists x_1\ ((\textbf{man'}\ x_1)\ \&\ ((\textbf{loves'}\ x_0)\ x_1))))$
= (by definition of interpretation **B** in (53) of previous handout)

(v) The function A such that if g ∈ J, A(g) = 1 *iff* for all for all x ∈ $D_{e,E}$,
if *k*(x) = 1 then there exists a y ∈ $D_{e,E}$ such that *i*(y) = 1 and *j*(x)(y)
= (by definition of interpretation **B** in (53) of previous handout)

(vi) The function A such that if g ∈ J, A(g) = 1

<u>Note:</u>
In our induced interpretation of Mini-English, tree (40a) – the 'inverse-scope reading of *some man loves every president*' – is interpreted as being true (relative to any variable assignment).

(68) **Illustration of the Induced Interpretation of Mini-English+Q**

Let T_2 be the tree for *some man loves every president* in (40b) [some > every].

(i) $h^\circ k(T_2)$ = (by definition of composition)

(ii) $h(k(T_2))$ = (by (66))

(iii) $h(((\lambda P_0\ \exists x_0\ ((\textbf{man'}\ x_0)\ \&\ (P_0\ x_0)))$
$(\lambda x_5\ ((\lambda P_0\ \forall x_0\ ((\textbf{president'}\ x_0) \rightarrow (P_0\ x_0)))\ (\lambda x_7\ ((\textbf{loves'}\ x_7)\ x_5)))))$ = (by (66))

(iv) $h(\exists x_0\ ((\textbf{man'}\ x_0)\ \&\ \forall x_1\ ((\textbf{president'}\ x_1) \rightarrow ((\textbf{loves'}\ x_1)\ x_0))))$ = (by def. of **B**)

(v) The function E such that if g ∈ J, *E*(g) = 1 iff there is an x ∈ $D_{e,E}$ such that
i(x) = 1 and for all y ∈ $D_{e,E}$, if *k*(y) = 1 then *j*(y)(x) = 1 = (by def. of **B**)

(vi) The function E such that if g ∈ J, A(g) = 1

Note:
In our induced interpretation of Mini-English, tree (40b) – the 'surface-scope reading of *some man loves every president*' – is interpreted as being true (relative to any variable assignment).

7. Summary

In these notes, we've accomplished the following:

- Developed a fragment of English (ME+Q) which contains quantificational terms (quantificational NPs)
- Provided a translation base homomorphically mapping the expressions of DME+Q to expressions of Politics+λ.
- Via our interpretation for Politics+λ defined in the last handout, obtained an 'induced' interpretation for ME+Q.

The resulting system has the following advantageous properties:

- Interprets sentences where quantificational terms are in direct object position
- Correctly / automatically predicts quantifier scope ambiguities in sentences containing more than one quantifier.
- Correctly / automatically predicts quantificational binding of pronouns (HOMEWORK!)

What's Next on the Agenda?

- Thus far, all our syntactic and semantic analyses have been within framework as presented in Montague's paper "Universal Grammar."
- In his paper "PTQ", however, Montague employs a relatively simplified presentation of the system, one that allows for a much more transparent / readable treatment of *opaque / intensional contexts*.
- Therefore, to build towards that, I will next 'transform' the system we developed in the last few handouts into a system akin to that presented in PTQ.
    - Again, the substance of the analysis will remain the same (as you'll see)
    - All that really differs is the notation / presentation employed…