

1. Lambda-Abstraction

Whereas first-order logic admits variables only of type D [individual-variables], type-theory admits variables of *every* type. Indeed, every expression of the form

$$\lambda v \mathcal{E}$$

is admitted, so long as v is a variable (*no matter* what its type), and \mathcal{E} is a well-formed expression (*no matter* what its type). The exact syntactic-type of $\lambda v \mathcal{E}$ is provided by the following general rule.

if v has type	\mathfrak{I}_1
and \mathcal{E} has type	\mathfrak{I}_2
then $\lambda v \mathcal{E}$ has type	$\mathfrak{I}_1 \rightarrow \mathfrak{I}_2$
Alternatively:	
$\text{type}(\lambda v \mathcal{E})$	$= \text{type}(v) \rightarrow \text{type}(\mathcal{E})$

The following is the attendant formatting rule.

If \mathcal{E} has type \mathfrak{I} , then

$[\lambda v \mathcal{E}]\langle \sigma \rangle$	is well-formed, and has type \mathfrak{I}	if $\text{type}(\sigma) = \text{type}(v)$
	is ill-formed	otherwise

2. Lambda-Conversion

The lambda-calculus has one very general axiom (schema), known as *lambda-conversion*, given as follows.

$\forall v \{ [\lambda v \mathcal{E}]\langle v \rangle = \mathcal{E} \}$	[lambda-conversion axiom]
--	---------------------------

Here, v is a variable, and \mathcal{E} is an expression, and ‘=’ is a generalized-identity predicate, which authorizes substitution in all contexts (in the style of Leibniz's Law).

The following is the corresponding rule, which is obtained by universal-instantiation.

$[\lambda v \mathcal{E}]\langle \sigma \rangle$	$= \mathcal{E}[\sigma/v]$	[lambda-conversion rule]
---	---------------------------	--------------------------

Here, $\text{type}(\sigma) = \text{type}(v)$, and v is free for σ in \mathcal{E} . As usual, $\mathcal{E}[\sigma/v]$ is, by definition, the result of substituting σ for every occurrence of v that is free in \mathcal{E} . To say that v is **free for** σ in \mathcal{E} is to say that every variable that is free in \mathcal{E} is also free in $\mathcal{E}[\sigma/v]$. In other words, no free variable gets accidentally-bound.

Examples

$$\begin{aligned}
 [\lambda x \mathbf{F}x]\langle a \rangle &= \mathbf{F}a \\
 [\lambda x \mathbf{R}xa]\langle a \rangle &= \mathbf{R}aa \\
 [\lambda x \mathbf{R}xx]\langle a \rangle &= \mathbf{R}aa \\
 [\lambda x \forall y \mathbf{R}xy]\langle a \rangle &= \forall y \mathbf{R}ay \\
 [\lambda P \forall x Px]\langle \lambda y \mathbf{F}y \rangle &= \forall x [\lambda y \mathbf{F}y]\langle x \rangle \\
 &= \forall x \mathbf{F}x \\
 [\lambda P \exists x Px]\langle \lambda y \mathbf{R}ya \rangle &= \exists x [\lambda y \mathbf{R}ya]\langle x \rangle \\
 &= \exists x \mathbf{R}xa \\
 [\lambda P \lambda Q \forall x \{ Px \rightarrow Qx \}]\langle \lambda y \mathbf{R}ya \rangle &= \lambda Q \forall x \{ [\lambda y \mathbf{R}ya]\langle x \rangle \rightarrow Qx \} \\
 &= \lambda Q \forall x \{ \mathbf{R}xa \rightarrow Qx \}
 \end{aligned}$$

3. Translating English into Type-Theory

1. Note 1

Before we continue, we note that the *standard* language of Type-Theory, like first-order logic, is systematically impoverished. In particular:

- (1) it lacks case-markers;
- (2) it treats common-nouns as one-place predicates;
- (3) it lacks an autonomous copula-be [alternatively, $\llbracket \text{be (cop)} \rrbracket = \emptyset$];
- (4) it lacks an autonomous indefinite-article [alternatively, $\llbracket \text{a} \rrbracket = \emptyset$].

For this reason, our translations are not completely natural, as indicated in the lexicon below.

2. Note 2

Henceforth, we use bolded lower-case and small-caps letters as proper-expressions of type D [i.e., proper-names/nouns], and we use lower-case math-italic letters as variables of type D [i.e., individual-variables]. Also, we use bolded upper-case letters as *proper-expressions* of type $D^* \rightarrow S$, and we use un-bolded upper-case letters as *variables* of type $D^* \rightarrow S$. Finally, we use upper-case Greek letters for *variables* of type S [there are no proper-expressions of type S].

3. Sample Lexicon

morpheme	type	translation	glossary
Jay, Kay, Elle	D	J, K, L	Jay, Kay, Elle
respects	$D \rightarrow D \rightarrow S$	$\lambda y \lambda x \mathbf{R}xy$	$\mathbf{R}[\alpha, \beta] =: \alpha$ respects β
admires		$\lambda y \lambda x \mathbf{A}xy$	$\mathbf{A}[\alpha, \beta] =: \alpha$ admires β
is (transitive)		$\lambda y \lambda x [x=y]$	
person is-a-person	\mathbf{C} [$=_{df} D \rightarrow S$]	$\lambda x \mathbf{P}x$	$\mathbf{P}[\alpha] =: \alpha$ is a person
man is-a-man		$\lambda x \mathbf{M}x$	$\mathbf{M}[\alpha] =: \alpha$ is a man
woman is-a-woman		$\lambda x \mathbf{W}x$	$\mathbf{W}[\alpha] =: \alpha$ is a woman
happy is-happy		$\lambda x \mathbf{H}x$	$\mathbf{H}[\alpha] =: \alpha$ is happy
virtuous is-virtuous		$\lambda x \mathbf{V}x$	$\mathbf{V}[\alpha] =: \alpha$ is virtuous
the-mother-of 's-mother (def)	$D \rightarrow D$	$\lambda x \{ \mathbf{m}(x) \}$	$\mathbf{m}(\alpha) =: \text{the mother of } \alpha$
brother-of 's-brother (indef) is-a-brother-of	$D \rightarrow D \rightarrow S$	$\lambda y \lambda x \mathbf{B}xy$	$\mathbf{B}[\alpha, \beta] =: \alpha$ is a brother of β
that, who	$D \rightarrow S. \rightarrow C \rightarrow C$	$\lambda P \lambda Q \lambda x \{ Qx \ \& \ Px \}$	
[mod]	$D \rightarrow S. \rightarrow C \rightarrow C$	$\lambda P \lambda Q \lambda x \{ Qx \ \& \ Px \}$	
every	$C \rightarrow D \rightarrow S. \rightarrow S$	$\lambda P \lambda Q \forall x \{ Px \rightarrow Qx \}$	
some		$\lambda P \lambda Q \exists x \{ Px \ \& \ Qx \}$	
no		$\lambda P \lambda Q \sim \exists x \{ Px \ \& \ Qx \}$	
the	$C \rightarrow D$	$\lambda P \iota x Px$	
not	$S \rightarrow S$	$\lambda \Phi \sim \Phi$	
and, but	$S^2 \rightarrow S$	$\lambda \Phi \Psi \{ \Phi \ \& \ \Psi \}$	

4. Standard Categorical-Composition

$$\llbracket \phi_1 + \phi_2 \rrbracket = \llbracket \phi_1 \rrbracket \oplus \llbracket \phi_2 \rrbracket$$

$$\begin{aligned} \text{where } \alpha \oplus \beta &= [\alpha] \langle \beta \rangle && \text{if } \text{type}(\alpha) = \text{type}(\beta) \rightarrow \mathfrak{I}, \text{ for some } \mathfrak{I} \\ &= [\beta] \langle \alpha \rangle && \text{if } \text{type}(\beta) = \text{type}(\alpha) \rightarrow \mathfrak{I}, \text{ for some } \mathfrak{I} \\ &= \emptyset && \text{otherwise} \end{aligned}$$

Here,

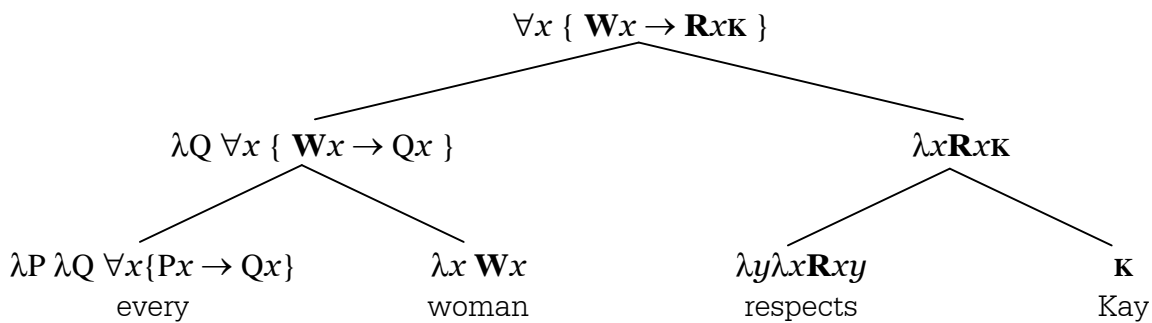
$$\begin{aligned} + &=_{df} \text{ syntactic composition} \\ \oplus &=_{df} \text{ semantic composition} \\ \llbracket \alpha \rrbracket &=_{df} \text{ the translation/meaning of } \alpha \end{aligned}$$

In other words, one can combine two phrases precisely if one serves as a type-appropriate argument for the other, in which case one obtains the composed meaning by applying the functor to the argument in accordance with lambda-conversion.

5. Examples

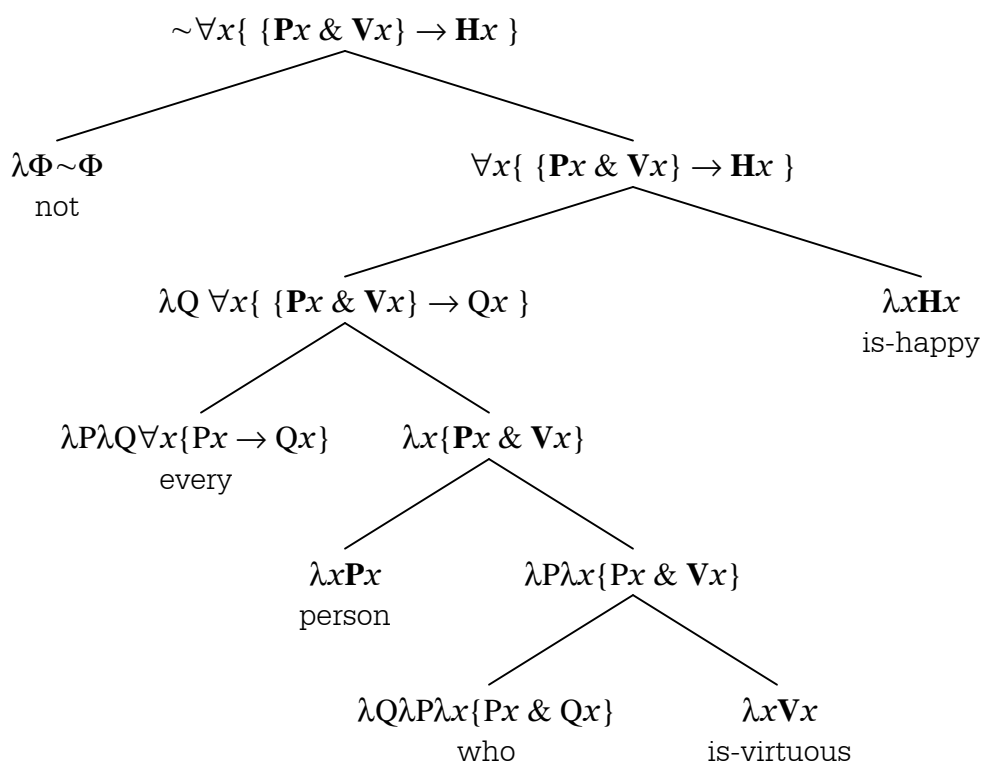
1. Example 1

every woman respects Kay



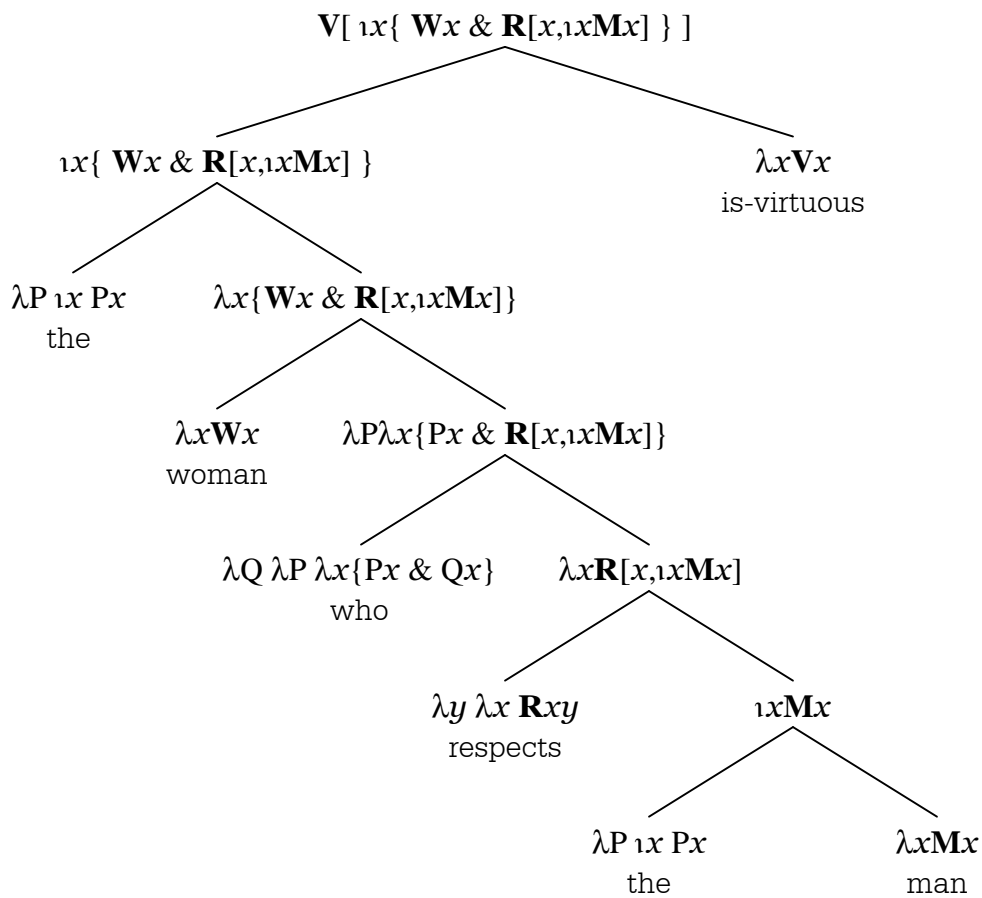
2. Example 2

not every person who is virtuous is happy



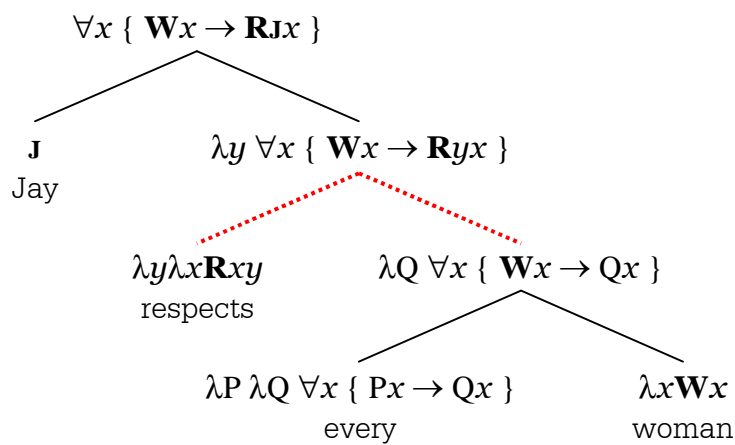
3. Example 3

the woman who respects the man is virtuous



4. Example 4 – Standard Composition Does not Work

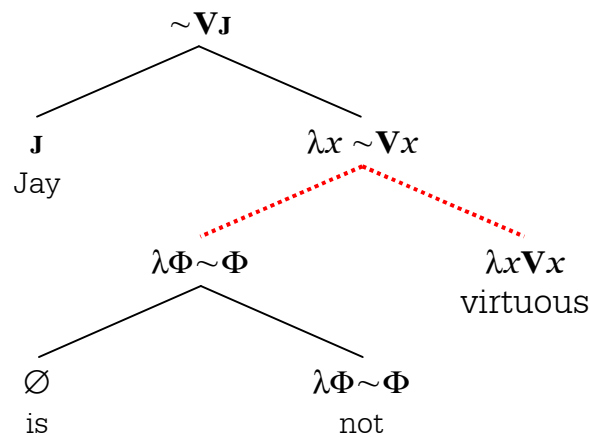
Jay respects every woman



In the above tree, \llbracket respects \rrbracket cannot be combined with \llbracket every woman \rrbracket by standard-composition, since neither phrase serves as a type-appropriate argument for the other.

5. Example 5 – Standard Composition Does not Work

Jay is not virtuous



In the above tree, even supposing that $\llbracket \text{is} \rrbracket$ and $\llbracket \text{not} \rrbracket$ compose as suggested, $\llbracket \text{is not} \rrbracket$ and $\llbracket \text{virtuous} \rrbracket$ do not compose properly, since neither phrase serves as a type-appropriate argument for the other.

4. Expanded Categorical Semantics

Just as we did in categorial syntax, we enlarge the permitted compositions in accordance with categorial logic. Note, for the sake of simplicity, for the moment, we consider only binary compositions.

\mathcal{E}_1 composes with \mathcal{E}_2 to produce \mathcal{E}_3 precisely if $\mathcal{E}_1 ; \mathcal{E}_2 \vdash \mathcal{E}_3$
 which is to say that there is a derivation of \mathcal{E}_3 from $\{\mathcal{E}_1, \mathcal{E}_2\}$ in the associated **categorial logic**.

5. Categorial Logic – Semantic Version

1. Definition of Semantic-Derivation

A semantic-derivation of \mathcal{E}_3 from $\{\mathcal{E}_1, \mathcal{E}_2\}$ is a sequence of lines as follows.

line number	expression	type	index	annotation
$?_1$	\mathcal{E}_1	$\mathfrak{I}(\mathcal{E}_1)$	m_1	Pr
$?_2$	\mathcal{E}_2	$\mathfrak{I}(\mathcal{E}_2)$	m_2	Pr
...
.	\mathcal{E}_3	$\mathfrak{I}(\mathcal{E}_3)$	m_1, m_2	...

In particular:

- (1) the first two lines are the premises – $\mathcal{E}_1, \mathcal{E}_2$.
- (2) every remaining line is either
 - (1) an assumption, or
 - (2) follows from previous lines by an inference-rule.
- (3) the last line is \mathcal{E}_3 .

2. The Premise-Rule

expression	type	index	annotation
\mathcal{E}	$\mathfrak{J}(\mathcal{E})$	m (new)	Pr

Here, \mathcal{E} is any expression of the λ -calculus, and m is any numeral that is new, which is to say it does not occur earlier in the derivation. Generally, we write all the premises first, and number the indices starting with 1.

3. The Assumption-Rule

expression	type	index	annotation
v (new)	$\mathfrak{J}(v)$	m (new)	As

Here, v is any variable of the λ -calculus that is new, which is to say it does not occur *unbound* earlier in the derivation, and m is any numeral that is new, which is to say it does not occur earlier in the derivation.

4. Lambda-Out (l O) [arrow-out]

line number	expression	type	index	annotation	restriction
$?_1$	Φ	$\mathcal{A} \rightarrow \mathcal{C}$	i	?	$i \not\subseteq j$ and $j \not\subseteq i$
$?_2$	\mathcal{E}	\mathcal{A}	j	?	
$?_3$	$[\Phi]\langle \mathcal{E} \rangle$	\mathcal{C}	$i+j$	$?_1, ?_2, \lambda O$	

Here, $[\Phi]\langle \mathcal{E} \rangle$ is the result of applying functor Φ to argument \mathcal{E} .

5. Lambda-In (l I) [arrow-in]

line number	expression	type	index	annotation
$?_1$	v (new)	\mathcal{A}	m (new)	As
$?_2$	\mathcal{E}	\mathcal{C}	$i+m$?
$?_3$	$\lambda v\{\mathcal{E}\}$	$\mathcal{A} \rightarrow \mathcal{C}$	i	$?_1, ?_2, \lambda I$

6. Lambda-Calculus (l C)

At any point in a derivation, one can apply any *equivalence* of the lambda-calculus to any expression. This includes most prominently applications of lambda-conversion.

7. Sub-Structural (Index) Rules

Like System R, except that the empty index \emptyset is forbidden. In particular,

- + is associative: $(i+j)+k = i+(j+k)$
- + is commutative: $i+j = j+i$
- + is contractive: $i+i = i$

6. Examples

1. Transitivity

The following is an example of a valid composition, in which A, B, and C are arbitrary types, and we (temporarily) pretend that ‘x’ is a variable of type A.

$$B \rightarrow C ; A \rightarrow B \vdash A \rightarrow C$$

(1)	P	$B \rightarrow C$	1	Pr
(2)	Q	$A \rightarrow B$	2	Pr
(3)	x	A	3	As
(4)	Q(x)	B	23	2,3, λO
(5)	P(Q(x))	C	123	1,4, λO
(6)	$\lambda x \mathbf{P(Q(x))}$	$A \rightarrow C$	12	3,5, λI

The following is an example of transitivity in which we compose $\llbracket \text{is not} \rrbracket$ and $\llbracket \text{virtuous} \rrbracket$ to obtain $\llbracket \text{is not virtuous} \rrbracket$.

(1)	$\lambda \Phi \sim \Phi$	$\llbracket \text{is not} \rrbracket$	$S \rightarrow S$	1	Pr
(2)	$\lambda x \mathbf{V}x$	$\llbracket \text{virtuous} \rrbracket$	$D \rightarrow S$	2	Pr
(3)	x	D	D	3	As
(4)	$[\lambda x \mathbf{V}x]\langle x \rangle$	S	S	23	2,3, λO
	Hx				λC
(5)	$[\lambda \Phi \sim \Phi]\langle \mathbf{V}x \rangle$	S	S	123	1,4, λO
	$\sim Hx$				λC
(6)	$\lambda x \sim \mathbf{V}x$	$\llbracket \text{is not virtuous} \rrbracket$	$D \rightarrow S$	12	3,5, λI

2. Accusative QPs

Earlier, we were unable to combine $\llbracket \text{respects} \rrbracket$ and $\llbracket \text{every woman} \rrbracket$ because of a type mismatch. The following derivation demonstrates how the appropriate meaning may be obtained using categorial logic.

$$\lambda z \lambda x \mathbf{R}xz ; \lambda Q \forall y \{ \mathbf{W}y \rightarrow \mathbf{Q}y \} \vdash \lambda x \forall y \{ \mathbf{W}y \rightarrow \mathbf{R}xy \}$$

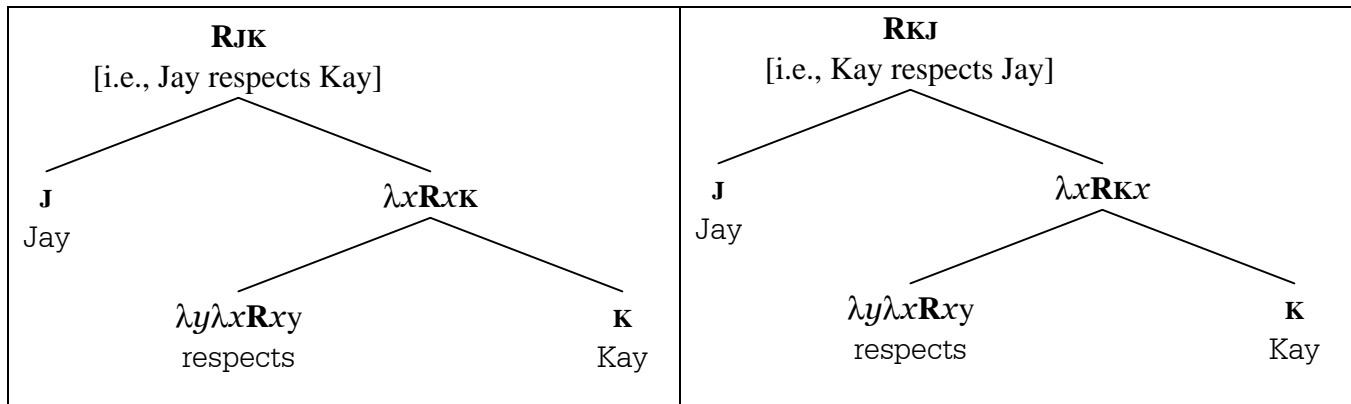
(1)	$\lambda y \lambda x \mathbf{R}xy$	$\llbracket \text{respects} \rrbracket$	$D \rightarrow (D \rightarrow S)$	1	Pr
(2)	$\lambda Q \forall y \{ \mathbf{W}y \rightarrow \mathbf{Q}y \}$	$\llbracket \text{every woman} \rrbracket$	$(D \rightarrow S) \rightarrow S$	2	Pr
(3)	x	D	D	3	As
(4)	z	D	D	4	As
(5)	$[\lambda z \lambda x \mathbf{R}xz]\langle z \rangle$	$D \rightarrow S$	$D \rightarrow S$	14	1,4, λO
	$\lambda x \mathbf{R}xz$				λC
(6)	$[\lambda x \mathbf{R}xz]\langle x \rangle$	S	S	134	3,5, λO
	$\mathbf{R}xz$				λC
(7)	$\lambda z \mathbf{R}xz$	$D \rightarrow S$	$D \rightarrow S$	13	4,6, λI
(8)	$[\lambda Q \forall y \{ \mathbf{W}y \rightarrow \mathbf{Q}y \}]\langle \lambda z \mathbf{R}xz \rangle$	S	S	123	2,7, λO
	$\forall y \{ \mathbf{W}y \rightarrow [\lambda z \mathbf{R}xz]\langle y \rangle \}$				λC
	$\forall y \{ \mathbf{W}y \rightarrow \mathbf{R}xy \}$				λC
(9)	$\lambda x \forall y \{ \mathbf{W}y \rightarrow \mathbf{R}xy \}$	$\llbracket \text{R's every W} \rrbracket$	$D \rightarrow S$	12	3,8, λI

7. A Fatal Flaw

Categorial logic provides very powerful tools for combining meanings, but unfortunately it is *way* too powerful, as it stands. In particular, both of the following compositions of $\llbracket \text{respects} \rrbracket$ and $\llbracket \text{Kay} \rrbracket$ are authorized.

$$\begin{aligned} \lambda y \lambda x \mathbf{R}xy ; \mathbf{K} &\vdash \lambda x \mathbf{R}x\mathbf{K} \\ \lambda y \lambda x \mathbf{R}xy ; \mathbf{K} &\vdash \lambda x \mathbf{R}\mathbf{K}x \end{aligned}$$

This means that both of the following semantic-compositions are admissible.



In other words, the semantics predicts that one reading of ‘Jay respects Kay’ is *that Kay respects Jay!* What is worse, one can generate indefinitely-many examples just like this; for example – one reading of ‘every dog is a mammal’ is *that every mammal is a dog*. This is a **complete disaster!**

8. Case-Markers to the Rescue

In order to correct the serious flaw discovered in the previous section, we propose to enlarge categorial semantics by including case-markers, just as we did in categorial syntax. In particular, when we combine ‘Kay’ with ‘respects’, we must also indicate whether ‘Kay’ is the subject or the object of the verb. This is exactly what case-marking is designed to do. As noted earlier, different languages mark cases/roles in different ways, including inflection, preposition, postposition, and word-order. For example, whereas Japanese uses postpositions, English uses a combination of the other three techniques. For example, in English, the subject-role is generally marked simply by placing the NP in front of the verb, and the object-role is generally marked simply by placing the NP after the verb.