Elizabeth Coppock
coppock@phil.hhu.de
Wed. November 2nd, 2011
Time: 14:30–16:00

Compositional Semantics
Heinrich Heine University
Winter Semester 2011/12
Room: 25.22-U1.72

Schönfinkelization, Types, and Lambdas

1 Functional application

Frege's conjecture: All semantic composition is functional application.

Functional application: The process of applying a function to an argument. We denote the result of functional application with parentheses: $f(x)$, read "f applied to x" or "f of x" is the unique y such that $\langle x, y\rangle \in f$. (Heim and Kratzer 1998, p. 10)

E.g. if EURO is the function that maps every country to true or false depending on whether it is on the Euro, then EURO(Germany) = 1, EURO(Sweden) = 0, etc. (Note that EURO is the *characteristic function* of the set of countries on the Euro.)

Functions that return functions. The result of applying a function can be another function. For example, we could have a function that maps currencies to functions that map countries to truth values. Suppose that CURRENCY is such a function. CURRENCY(Euro) would be the same as EURO, so CURRENCY(Euro) applied to Germany would return 1:

[CURRENCY(Euro)](Germany) = EURO(Germany) = 1
[CURRENCY(Euro)](Sweden) = EURO(Sweden) = 0
[CURRENCY(Dollar)](Germany) = [CURRENCY(Dollar)](Sweden) = 0.[1]

How do we define a function that returns another function? We don't want to name every output function. One way is to list all the input and output values.

$$\begin{bmatrix} \text{Euro} & \to & \begin{bmatrix} \text{Sweden} & \to & 0 \\ \text{Germany} & \to & 1 \end{bmatrix} \\ \text{Dollar} & \to & \begin{bmatrix} \text{Sweden} & \to & 0 \\ \text{Germany} & \to & 0 \end{bmatrix} \end{bmatrix}$$

[1] The square brackets can be omitted when it is clear enough what goes with what; CURRENCY(Euro)(Sweden) = [CURRENCY(Euro)](Sweden).

But what if we want to do it "by description" instead? Use λ notation.

CURRENCY = $[\lambda m$. $[\lambda c$. c uses the m as a currency]]

So, [CURRENCY(Euro)](Germany)
= [[λm . [λc . c uses the m as a currency]](Euro)](Germany)
= [λc . c uses the Euro as a currency](Germany)
= Germany uses the Euro as a currency
= 1

[λm . [λc . c uses the m as a currency](Germany)](Euro)
= [λm . Germany uses the m as a currency](Euro)
= Germany uses the Euro as a currency
= 1

2 A fragment of English (H&K 1998, ch. 2)

A. Inventory of denotations

(i) Elements of D, the set of actual individuals
(ii) Elements of $\{0,1\}$, the set of truth values
(iii) Functions from D to $\{0,1\}$

B. Lexicon

⟦Ann⟧	=	Ann	
⟦Jan⟧	=	Jan	
⟦works⟧	=	f :	$D \to \{0,1\}$
			For all $x \in D$, $f(x)$ = 1 iff x works
	=	$\lambda x : x \in D$. x works	
⟦smokes⟧	=	f :	$D \to \{0,1\}$
			For all $x \in D$, $f(x)$ = 1 iff x smokes
	=	$\lambda x : x \in D$. x smokes	

If Ann, Jan and Maria are the only individuals in the domain (so D = {Ann, Jan, Maria}), and Ann and Jan are the only smokers, then:

$$\textbf{⟦smokes⟧} = \begin{bmatrix} \text{Ann} & \to & 1 \\ \text{Jan} & \to & 1 \\ \text{Maria} & \to & 0 \end{bmatrix}$$

C. Composition rules

(S1) If α has the form 

then $[\![\alpha]\!] = [\![\gamma]\!]([\![\beta]\!])$.

(S2) If α has the form 

then $[\![\alpha]\!] = [\![\beta]\!]$.

(S3) If α has the form 

then $[\![\alpha]\!] = [\![\beta]\!]$.

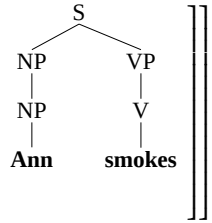
(S4) If α has the form 

then $[\![\alpha]\!] = [\![\beta]\!]$.

(S5) If α has the form 

then $[\![\alpha]\!] = [\![\beta]\!]$.

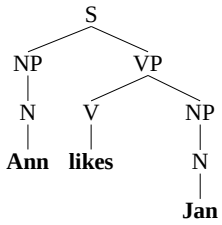
Claim:

$[\![$ 

$]\!]$ = 1 iff Ann smokes.

3 Schönfinkelization

Semantics for transitive verbs. Example:



To match rule (S1), the VP should be a function from individuals to truth values (type $\langle e,t\rangle$).

Aside: Notation for semantic types

- e is the type of individuals; $D_e = D$
- t is the type of truth values; $D_t = \{0,1\}$

A function has type $\langle e,t\rangle$ if it is a function from individuals to truth values, etc.

- $D_{\langle e,t\rangle} = \{f : f$ is a function from D_e to $D_t\}$
- $D_{\langle e,\langle e,t\rangle\rangle} = \{f : f$ is a function from $D_{\langle e,t\rangle}$ to $D_t\}$

Semantic types are built up recursively:

- e and t are semantic types.
- If σ and τ are semantic types, then $\langle\sigma,\tau\rangle$ is a semantic type.
- Nothing else is a semantic type.

Back to the semantics of transitive verbs. We said before that the VP should be type $\langle e,t\rangle$. Since **Jan** denotes an individual, the semantic value of the transitive verb **likes** should be *a function from individuals to functions from individuals to truth values,* i.e, it should have the following type:

$$\langle e,\langle e,t\rangle\rangle$$

Here is an example. Suppose that Jan likes Ann, but nobody else likes anybody other than themselves (and everybody likes themselves). We can represent this as a set of ordered pairs this:

(1) $\{\langle \text{Jan}, \text{Ann}\rangle, \langle \text{Jan}, \text{Jan}\rangle, \langle \text{Ann}, \text{Ann}\rangle, \langle \text{Maria}, \text{Maria}\rangle\}$

Or we can specify the denotation of **likes** as a function of type $\langle e,\langle e,t\rangle\rangle$.

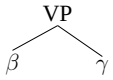
To do this, first we find the characteristic function of the liking relation:

(2)
$$\begin{bmatrix} \langle \text{Ann}, \text{Ann}\rangle & \to & 1 \\ \langle \text{Ann}, \text{Jan}\rangle & \to & 0 \\ \langle \text{Ann}, \text{Maria}\rangle & \to & 0 \\ \langle \text{Jan}, \text{Jan}\rangle & \to & 1 \\ \langle \text{Jan}, \text{Ann}\rangle & \to & 1 \\ \langle \text{Jan}, \text{Maria}\rangle & \to & 0 \\ \langle \text{Maria}, \text{Maria}\rangle & \to & 1 \\ \langle \text{Maria}, \text{Ann}\rangle & \to & 0 \\ \langle \text{Maria}, \text{Jan}\rangle & \to & 0 \end{bmatrix}$$

Then we Schönfinkel it right to left. This will result in a function F such that $[F(Y)](X)$ = 1 if and only if $R(X,Y)$.

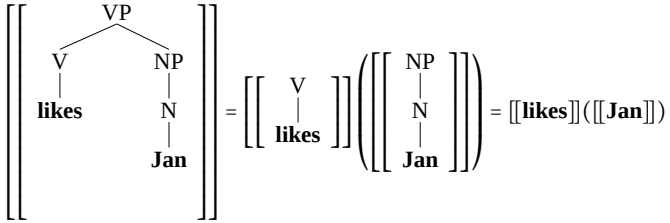
(3)
$$\begin{bmatrix} \text{Jan} \rightarrow \begin{bmatrix} \text{Jan} \rightarrow 1 \\ \text{Ann} \rightarrow 0 \\ \text{Maria} \rightarrow 0 \end{bmatrix} \\ \text{Ann} \rightarrow \begin{bmatrix} \text{Jan} \rightarrow 1 \\ \text{Ann} \rightarrow 1 \\ \text{Maria} \rightarrow 0 \end{bmatrix} \\ \text{Maria} \rightarrow \begin{bmatrix} \text{Jan} \rightarrow 0 \\ \text{Ann} \rightarrow 0 \\ \text{Maria} \rightarrow 1 \end{bmatrix} \end{bmatrix}$$

The right-to-left version gives us what we need for compositional interpretation of transitive verbs, because transitive verbs apply to the object first, and then the subject.

(S6) If α has the form 

then $[\![\alpha]\!] = [\![\gamma]\!]([\![\beta]\!])$.

So:

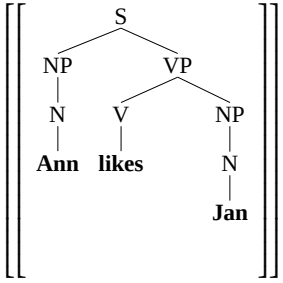


$= [\![\text{likes}]\!]([\![\text{Jan}]\!])$

And if $[\![\textbf{likes}]\!]$ is defined as in (3), then:

$$[\![\textbf{likes}]\!]([\![\textbf{Jan}]\!]) = \begin{bmatrix} \text{Jan} \rightarrow \begin{bmatrix} \text{Jan} \rightarrow 1 \\ \text{Ann} \rightarrow 0 \\ \text{Maria} \rightarrow 0 \end{bmatrix} \\ \text{Ann} \rightarrow \begin{bmatrix} \text{Jan} \rightarrow 1 \\ \text{Ann} \rightarrow 1 \\ \text{Maria} \rightarrow 0 \end{bmatrix} \\ \text{Maria} \rightarrow \begin{bmatrix} \text{Jan} \rightarrow 0 \\ \text{Ann} \rightarrow 0 \\ \text{Maria} \rightarrow 1 \end{bmatrix} \end{bmatrix} ([\![\textbf{Jan}]\!]) = \begin{bmatrix} \text{Jan} \rightarrow 1 \\ \text{Ann} \rightarrow 0 \\ \text{Maria} \rightarrow 0 \end{bmatrix}$$

This is the perfect type of meaning to be applied to a subject, so we get:



$= \begin{bmatrix} \text{Jan} \rightarrow 1 \\ \text{Ann} \rightarrow 0 \\ \text{Maria} \rightarrow 0 \end{bmatrix} ([\![\textbf{Ann}]\!]) = 0.$

Schönfinkelizing ternary relations. Ternary relations can also be Schönfinkelized. Imagine that Barack Obama and Angela Merkel are the only two individuals in the universe, and that there is a ternary relation that holds between three individuals X, Y and Z iff X told Y about Z. Barack Obama told Angela Merkel about himself, and Angela Merkel told Barack Obama about himself. This extension of this relation, call it R, can be given as a set of ordered triples:

{⟨ Barack, Angela, Barack ⟩,⟨ Angela, Barack, Barack ⟩}

The characteristic function of this set is as follows:

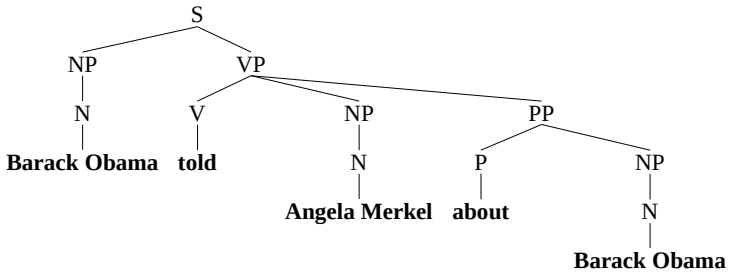
(4)
$$\begin{bmatrix} \langle \text{Barack}, \text{Barack}, \text{Barack} \rangle \rightarrow 0 \\ \langle \text{Barack}, \text{Barack}, \text{Angela} \rangle \rightarrow 0 \\ \langle \text{Barack}, \text{Angela}, \text{Barack} \rangle \rightarrow 1 \\ \langle \text{Barack}, \text{Angela}, \text{Angela} \rangle \rightarrow 0 \\ \langle \text{Angela}, \text{Barack}, \text{Barack} \rangle \rightarrow 1 \\ \langle \text{Angela}, \text{Barack}, \text{Angela} \rangle \rightarrow 0 \\ \langle \text{Angela}, \text{Angela}, \text{Barack} \rangle \rightarrow 0 \\ \langle \text{Angela}, \text{Angela}, \text{Angela} \rangle \rightarrow 0 \end{bmatrix}$$

Now we Schönfinkelize it from right-to-left, yielding a function F such that:

$[[F(Z)](Y)](X) = 1$ iff $R(X,Y,Z)$

(5)
$$\left[\begin{array}{ll} \text{Angela} \rightarrow & \left[\begin{array}{ll} \text{Barack} \rightarrow & \left[\begin{array}{ll} \text{Barack} \rightarrow & 0 \\ \text{Angela} \rightarrow & 0 \end{array}\right] \\ \text{Angela} \rightarrow & \left[\begin{array}{ll} \text{Barack} \rightarrow & 0 \\ \text{Angela} \rightarrow & 0 \end{array}\right] \end{array}\right] \\ \text{Barack} \rightarrow & \left[\begin{array}{ll} \text{Barack} \rightarrow & \left[\begin{array}{ll} \text{Barack} \rightarrow & 0 \\ \text{Angela} \rightarrow & 1 \end{array}\right] \\ \text{Angela} \rightarrow & \left[\begin{array}{ll} \text{Barack} \rightarrow & 1 \\ \text{Angela} \rightarrow & 0 \end{array}\right] \end{array}\right] \end{array}\right]$$

Suppose we have a phrase structure rule for ditransitive verbs like *tell*, that generates sentences like *X told Y about Z*. It could generate trees like this for example:



Can we write a semantic interpretation rule that sticks to Frege's Conjecture and uses F' as the denotation of **told**? Would it help if the PP were adjoined to the VP?

Lambda (λ) notation. Recall:

$$\begin{array}{lll} [\![\textbf{works}]\!] & = & f: \quad D \rightarrow \{0,1\} \\ & & \quad\quad \text{For all } x \in D, f(x) = 1 \text{ iff } x \text{ works} \\ & = & \lambda x : x \in D \ .\ x \text{ works} \end{array}$$

Generally λ-terms have the following schema:

$$\lambda\alpha : \phi \ .\ \gamma$$

where

- α is the argument variable (a letter that stands for an arbitrary argument of the function we are defining)
- ϕ is the domain condition (places a condition on possible values for α)

- γ is the value description (specifies the value that our function assigns to α)

It can be read, "the function which maps every α such that ϕ to 1 if γ and 0 otherwise", or: "the function which maps every α such that ϕ to 1 <u>iff</u> γ".

E.g. "the function which maps every x such that $x \in D$ to 1 iff x works."

The result of applying a function specified using λ notation to its argument is the value description part (γ), with the argument substituted for the argument variable (α).

$[\lambda x : x \in D \ .\ \underline{x \text{ smokes}}\ ]$(Ann) = Ann smokes.

Back to the semantics of transitive verbs (again).

(6) $[\![\textbf{likes}]\!] = \lambda x : x \in D.[\lambda y : y \in D \ .\ y \text{ likes } x]$

(7) $[\![\textbf{likes}]\!]([\![\textbf{Jan}]\!]) = [\lambda x : x \in D \ .\ [\lambda y : y \in D.y \text{ likes } x]]$(Jan)
= $\lambda y : y \in D \ .\ y$ likes Jan

(8) $[[\![\textbf{likes}]\!]([\![\textbf{Jan}]\!])][\![\textbf{Ann}]\!] = [\lambda y : y \in D \ .\ y$ likes Jan](Ann)
= Ann likes Jan

More fun with lambdas.

[λx $\in$D . [λy $\in$D . [λz $\in$D . z told x about y]]](Obama)(Merkel)
= [λy $\in$D . [λz $\in$D . z told Obama about y]](Merkel)
= [λz $\in$D . z told Obama about Merkel]]

[λx $\in$D . [λy $\in$D . [λz $\in$D . z told x about y](Obama)](Merkel)]
= [λx $\in$D . [λy $\in$D . Obama told x about y](Merkel)]
= [λx $\in$D . Obama told x about Merkel]

[λf $\in$D$_{\langle e,\langle e,t\rangle\rangle}$. [λx $\in$D$_e$. f(x)(Fred) = 1]]([λy $\in$D$_e$. [λz $\in$D$_e$. z kissed y]])
= [λx $\in$D$_e$. [λy $\in$D$_e$. [λz $\in$D$_e$. z kissed y]](x)(Fred) = 1]
= [λx $\in$D$_e$. [λz $\in$D$_e$. z kissed x](Fred) = 1]
= [λx $\in$D$_e$. Fred kissed x]