

Understanding the relationship between slashes and lambdas

It's important to understand that while slashes indicate missing pieces in syntax, and lambdas indicate missing pieces in semantics, *they're talking about the very same missing pieces*.

To see what we mean by this consider the following really simple derivation for 'John left':

John	left	
NP	S\NP	
<i>john'</i>	$\lambda x.left'(x)$	
S		<
<i>left'(john')</i>		

It's no coincidence that of the two pieces that make up this sentence, 'left' is the one that has a syntactic category with a slash in it and also the one that has a semantic value with a lambda on the front. It has to be this way. The category S\NP says "put an NP next to me and together we'll make up an S"; the semantic value $\lambda x.left'(x)$ says, basically, "give me an entity and together we'll make up the statement that that entity left". The entity which fills the empty slot in the semantic value *is* the semantic value of the NP which fills the empty slot in the category S\NP.

The general "recipe" for combining things in this way (the function application rule) is the following, copied from the definition in (13b) on page 13 of Steedman 1996 (chapter 2):

Y	X\Y	
<i>a</i>	<i>f</i>	
X		<
<i>f(a)</i>		

The thing that has a syntactic category which is looking for something (ie. a syntactic category with a slash in it) is the one which acts as a function, in the semantics. The thing that has the looked-for category is the one which acts as the argument to that function, in the semantics. In the concrete example above, the semantic value corresponding to *a* is *john'*, and the semantic value corresponding to *f* is $\lambda x.left'(x)$. The semantic result in the recipe is *f(a)*, which means applying *f* to *a*; for the concrete example, we need to apply $\lambda x.left'(x)$ to *john'*:

$$(\lambda x.left'(x)) (john')$$

So we're filling in the semantic empty slot indicated by the λx with *john'*: we get rid of the the λx on the front, since the slot is no longer empty, and anywhere else that *x* appears, we replace it with *john'*:

$$left'(john')$$

A more complicated example

In the following sentence let's say 'bet' is a verb which takes *four* arguments, the four bracketed pieces, not just one argument like 'left' did in the example above. Its four "empty slots" are indicated by the four coloured pieces of the syntactic category, and the four corresponding coloured pieces of its semantics¹:

[I]_{NP} bet [John]_{NP} [five dollars]_{NP} [that I would win the race]_S

bet
(((S\NP)/S)/NP)/NP
$\lambda w \lambda x \lambda y \lambda z. bet'(w)(x)(y)(z)$

There are four empty slots — we might think of them as the orange slot, the green slot, the blue slot and the red slot — and each one is represented in the syntax by a slash and in the semantics by a lambda. Notice that the empty slot represented at the *end* of the syntactic category (by the orange NP) is represented at the *beginning* of the semantics (by the orange λw).

This slot represented in orange is the first one that 'bet' is looking to fill. From the direction of the slash which introduces the orange NP, we know that the thing which is going to fill this slot must be to the right of 'bet'. 'John' will do nicely. So the first step of the derivation goes like this:

bet	John
(((S\NP)/S)/NP)/NP	NP
$\lambda w \lambda x \lambda y \lambda z. bet'(w)(x)(y)(z)$	$john'$
((S\NP)/S)/NP	
$\lambda x \lambda y \lambda z. bet'(john')(x)(y)(z)$	

Now the orange slot has been filled — we have only a green slot, a blue slot and a red slot left. The syntax (slashes) and the semantics (lambdas) both reflect this.

Of course, in order to arrive at the semantic value for the result of that step, what actually happened was we applied the original semantics for 'bet' to the semantics for 'John', as stated by the recipe²:

$$(\lambda w \lambda x \lambda y \lambda z. bet'(w)(x)(y)(z)) (john')$$

When we see this, we get rid of the orange λw , since this slot is being filled, and replace w with $john'$ anywhere else we find it, which results in $\lambda x \lambda y \lambda z. bet'(john')(x)(y)(z)$.

Now we have a big thing with category ((S\NP)/S)/NP, so the next thing it's looking for is another NP to its right to fill the green slot. The NP 'five dollars' is ready to fill that role. So the next step proceeds almost exactly like the first:

bet	John	
(((S\NP)/S)/NP)/NP	NP	
$\lambda w \lambda x \lambda y \lambda z. bet'(w)(x)(y)(z)$	$john'$	five dollars
((S\NP)/S)/NP		NP
$\lambda x \lambda y \lambda z. bet'(john')(x)(y)(z)$		$five-dollars'$
(S\NP)/S		
$\lambda y \lambda z. bet'(john')(five-dollars')(y)(z)$		

¹I hope these colours are distinguishable. I'm colour-blind, so I can only hope!

²Actually, here we're using the "forward-looking" version of the recipe, shown in (13a) on page 13 of Steedman 1996, rather than the "backward-looking" version used in the last example.

No more green slot. The only unfilled slots now are the blue slot and the red slot, and again the syntax (slashes) and the semantics (lambdas) both reflect this.

The next step is exactly the same as the previous two: we now have something looking rightward for something of category S, and ‘that I would win the race’ is waiting right there to fill that slot, the blue slot. The very last step, filling the red slot, is virtually the same again, the only difference being that the NP to fill that slot needs to be to the left. (Notice that this directionality is indicated only in the slashes, not in the lambdas.) So things finish up like this:

	$\frac{\text{bet}}{((S \backslash \text{NP}) / S) / \text{NP} / \text{NP}}$ $\lambda w \lambda x \lambda y \lambda z. \text{bet}'(w)(x)(y)(z)$	$\frac{\text{John}}{\text{NP}}$ john'	
	$\frac{((S \backslash \text{NP}) / S) / \text{NP}}{\lambda x \lambda y \lambda z. \text{bet}'(\text{john}')(x)(y)(z)}$	$\frac{\text{five dollars}}{\text{NP}}$ $\text{five-dollars}'$	$\frac{\text{that I would win the race}}{\text{S}}$ $\text{me-win-race}'$
$\frac{\text{I}}{\text{NP}}$ i'	$\frac{(S \backslash \text{NP}) / S}{\lambda y \lambda z. \text{bet}'(\text{john}')(\text{five-dollars}')(y)(z)}$	$\frac{\text{S} \backslash \text{NP}}{\lambda z. \text{bet}'(\text{john}')(\text{five-dollars}')(\text{me-win-race}')(z)}$	
	$\frac{\text{S}}{\text{bet}'(\text{john}')(\text{five-dollars}')(\text{me-win-race}')(i')}$		