Alternatives and binding

Computational semantics seminar

February 8, 2017

A quick announcement

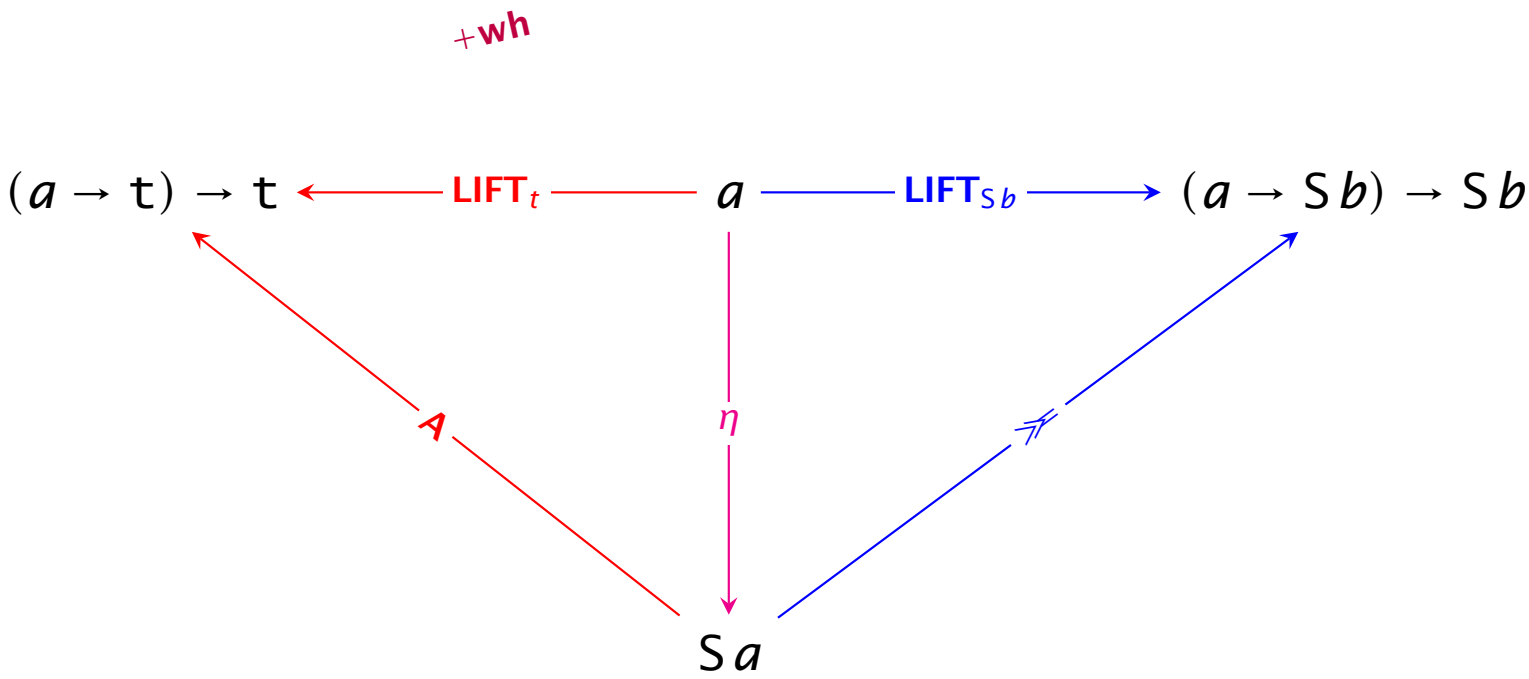
Your one-stop shop for readings, slides, and exercises:

`https://github.com/schar/comp-sem`

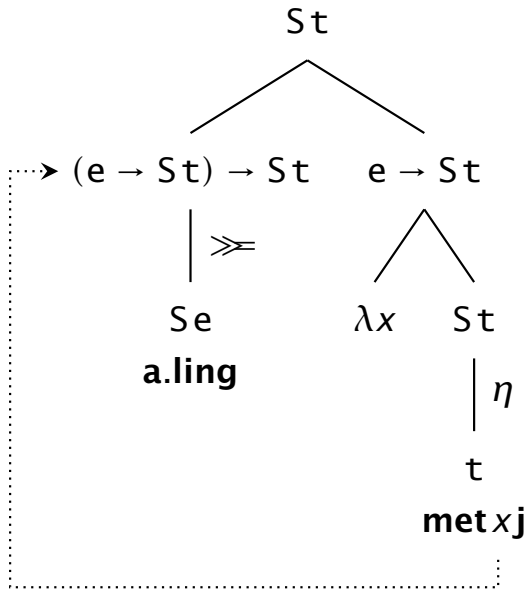
Some of the readings require a password. That password is `rutgers`.

Basics

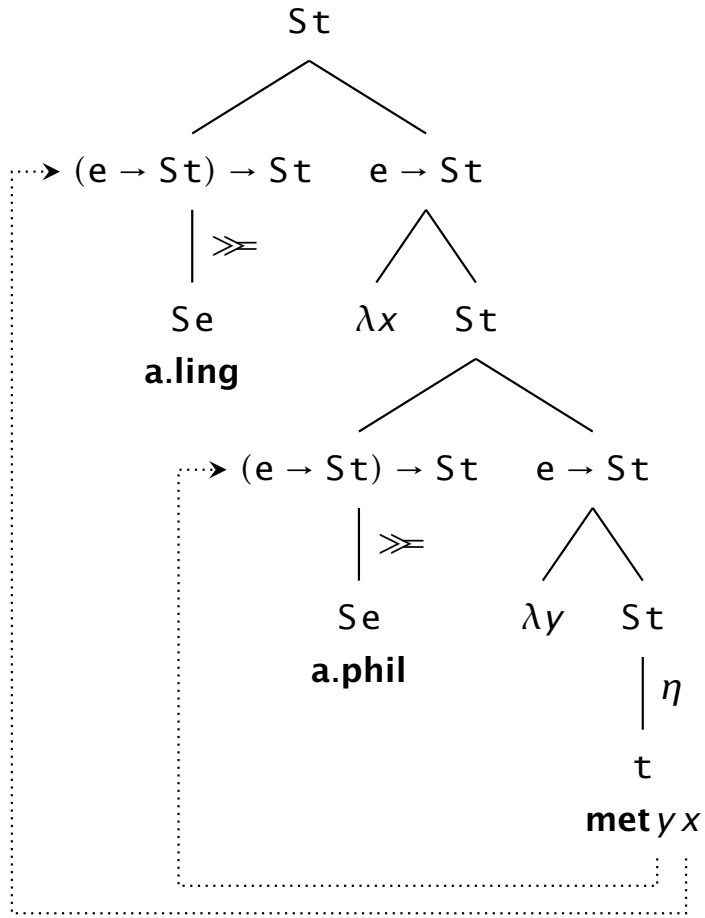
The Partee (1986) triangle++



Basic derivations



$= \{\mathbf{met}\, x\, \mathbf{j} \mid \mathbf{ling}\, x\}$



$= \{\mathbf{met}\, y\, x \mid \mathbf{ling}\, x, \mathbf{phil}\, y\}$

A reminder about scope

We aren't assuming that scope requires actual syntactic movement, which would in turn require some apparatus for binding a trace.

So we're theorizing about scope-taking separately from whatever apparatus underlies the interpretation of pronouns, traces, and binding.

This is perfectly ok to do! (As we'll see in detail later on in the seminar.) But it is somewhat non-standard, and so it's worth flagging.

Reminder: the all-encompassing $\gg=$

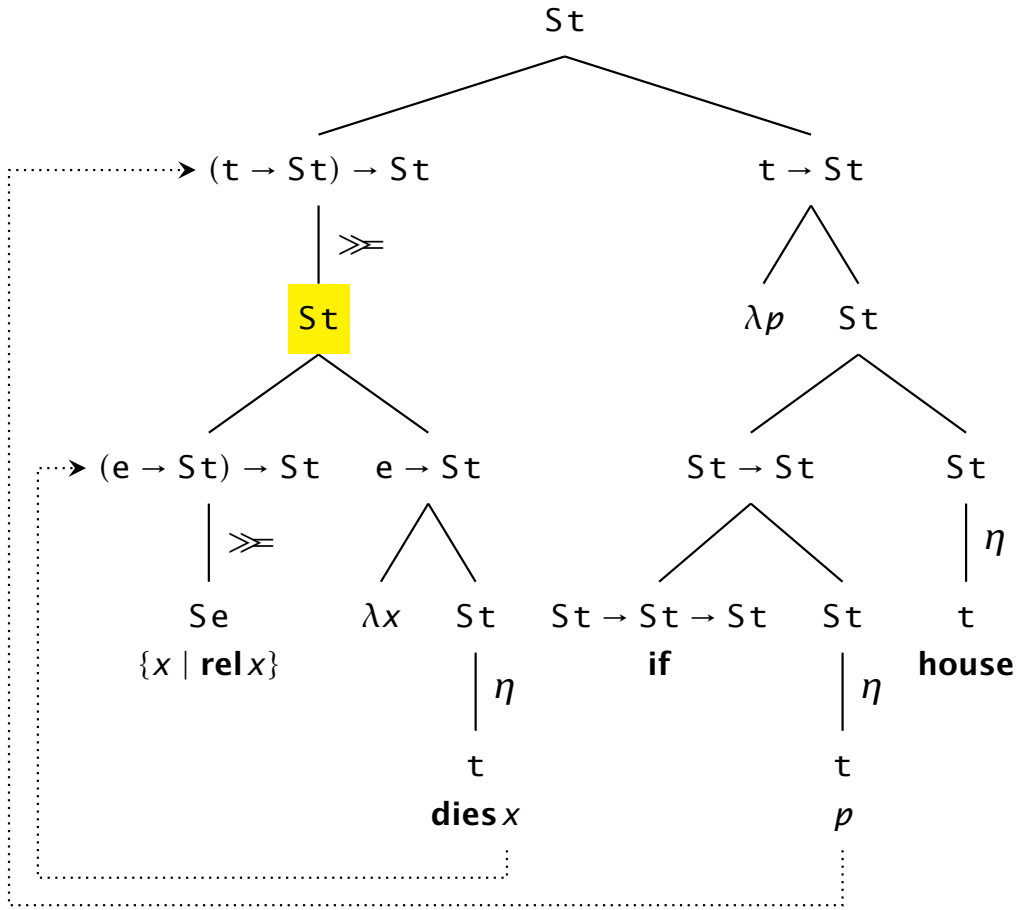
Remember that $\gg=$ is defined *polymorphically*. It can apply to any set of alternatives, whatsoever.

So, for example, it's possible to $\gg=$-shift an *island*!

$$\begin{aligned}\{\mathbf{dies}\,x \mid \mathbf{rel}\,x\}^{\gg=} &= \lambda f.\bigcup_{p\in\{\mathbf{dies}\,x\mid\mathbf{rel}\,x\}} f\,p \\ &= \lambda f.\bigcup_{x\in\mathbf{rel}} f\,(\mathbf{dies}\,x)\end{aligned}$$

The effect is as if the indefinite's alternatives have been sifted out! The thing fed to f is... the rest of the island!

A simple case of exceptional scope



$$= \{\textbf{dies}\, x \Rightarrow \textbf{house} \mid \textbf{rel}\, x\}$$

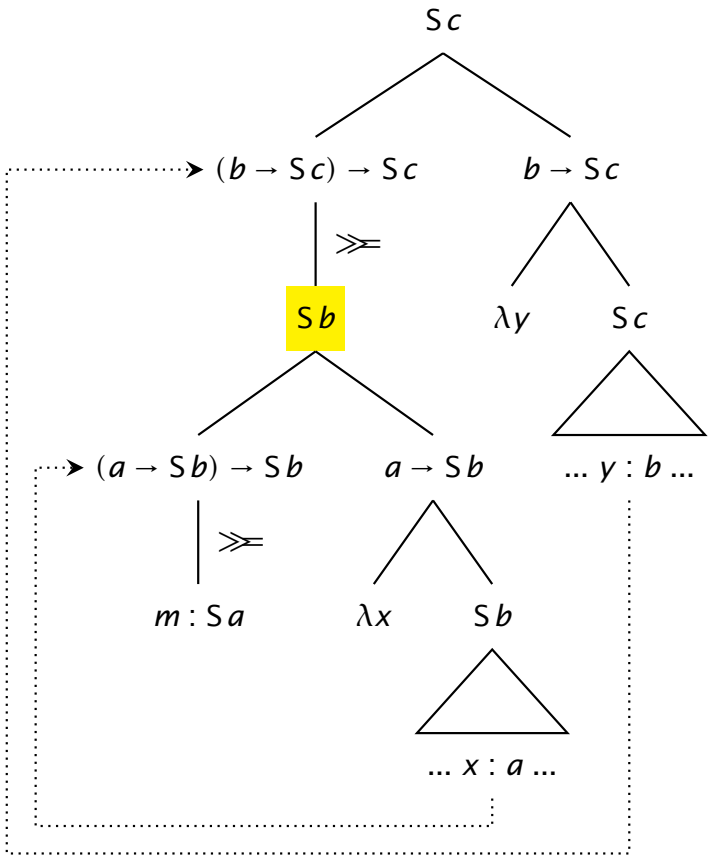
Reminder: Associativity of $\gg=$

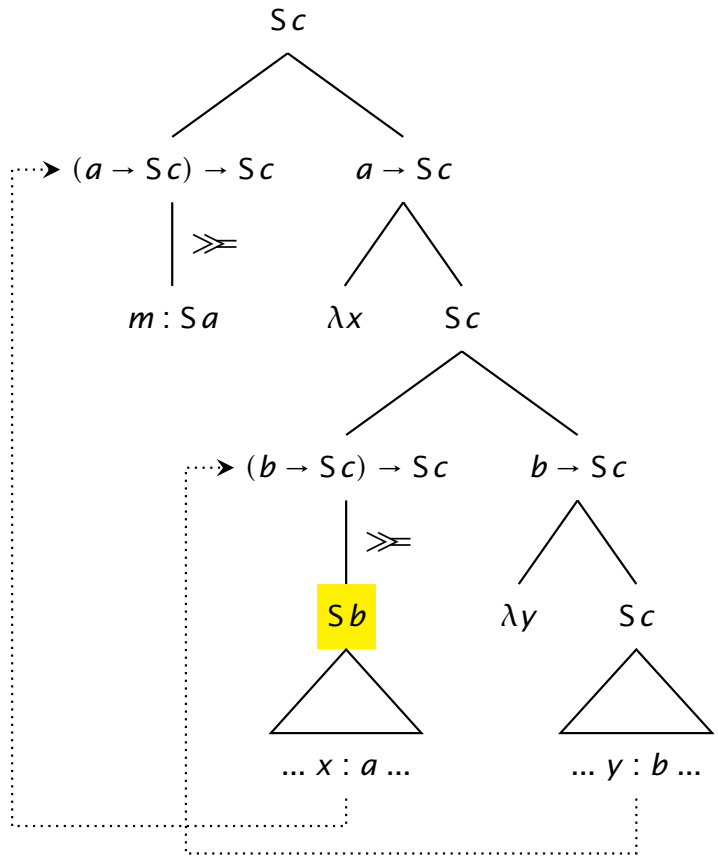
The reason this works is that $\gg=$ obeys a kind of **Associativity**:

$$(m \gg= \lambda x.\, f\, x) \gg= g = m \gg= (\lambda x.\, f\, x \gg= g)$$

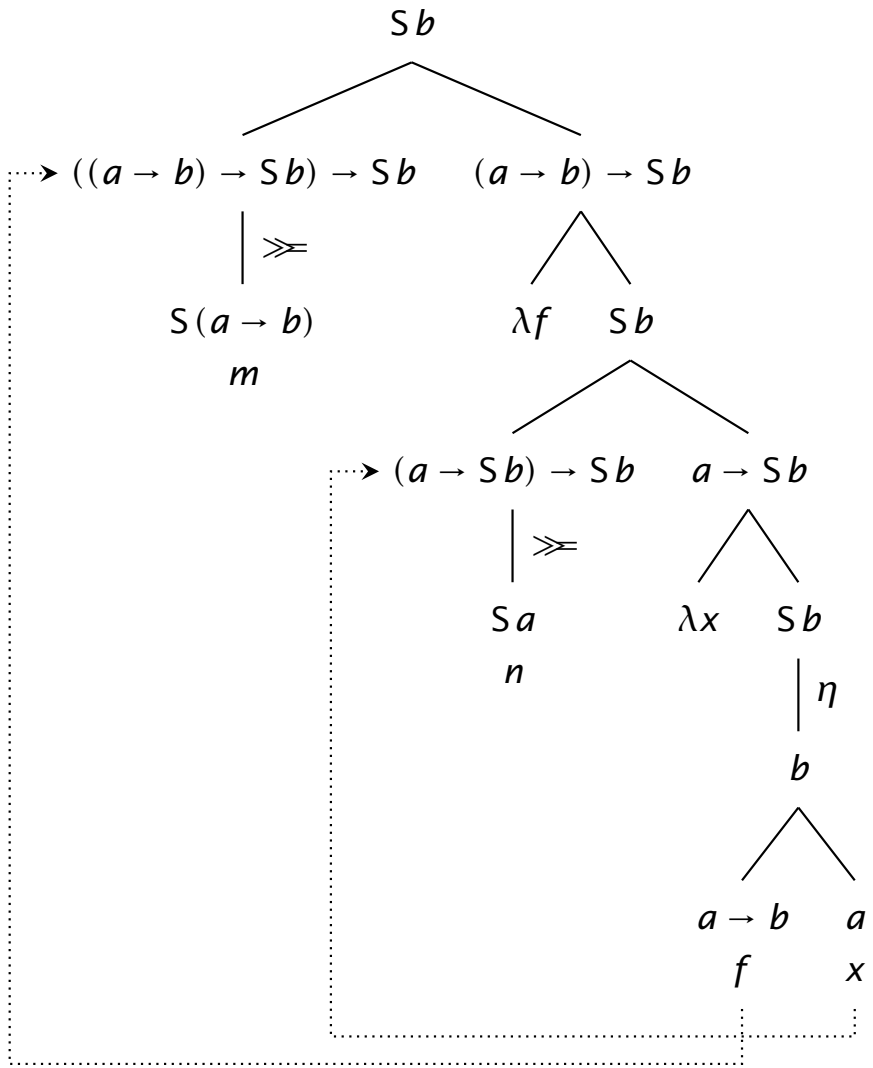
Associativity law in tree form

The import is clearer if we present **Associativity** in tree form:





A mysterious tree. . . Can you work out what's happening?



An ‘applicative’

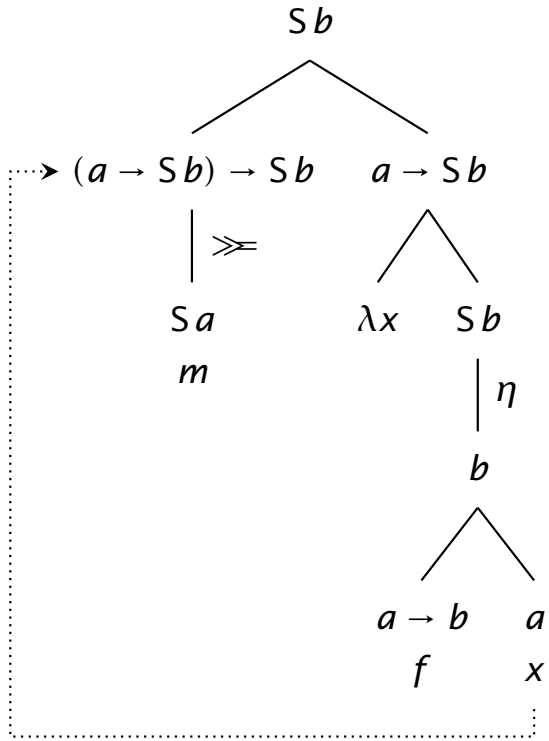
Two invocations of $\gg\!=$ (and one of η) **derives** the point-wise functional application operation that undergirds alternative semantics:

$$\begin{aligned} m \gg\!= \lambda f.\, n \gg\!= \lambda x.\, \eta\,(f\,x) &= m \gg\!= \lambda f.\, n \gg\!= \lambda x.\, \{f\,x\} \\ &= m \gg\!= \lambda f.\, \bigcup_{x \in n} \{f\,x\} \\ &= \bigcup_{f \in m} \bigcup_{x \in n} \{f\,x\} \\ &= \{f\,x \mid f \in m, x \in n\} \end{aligned}$$

But this operation is not stipulated as a default mode of composition. It ‘arises’ on its own.

More generally, as we will see, *every* monad gives rise to an enriched sense of functional application, and in an analogous way. The jargony short-hand for this: every monad is **applicative**.

Another mysterious tree... What's happening here?



A ‘functor’

Let’s see what that tree amounts to:

$$\begin{aligned} m \gg\!= \lambda x.\, \eta\,(f\,x) &= m \gg\!= \lambda x.\, \{f\,x\} \\ &= \bigcup_{x \in m} \{f\,x\} \\ &= \{f\,x \mid x \in m\} \end{aligned}$$

Does this remind you of anything?

A ‘functor’

Let’s see what that tree amounts to:

$$\begin{aligned} m \gg\!= \lambda x.\, \eta\,(f\,x) &= m \gg\!= \lambda x.\, \{f\,x\} \\ &= \bigcup_{x \in m} \{f\,x\} \\ &= \{f\,x \mid x \in m\} \end{aligned}$$

Does this remind you of anything? This is the **mapping** operation over sets, again derived via the monadic functions η and $\gg\!=$.

More generally, *every* monad gives rise to a mapping operation, and in an analogous way. The short-hand for this: every monad is a **functor**.

Breaking it down

As we begin to hint at in the last class (and in the exercises for this week!), an alternative way to formulate a monad is to break $\gg=$ down into two functions — the monad's mapping operation, and a **join** function:

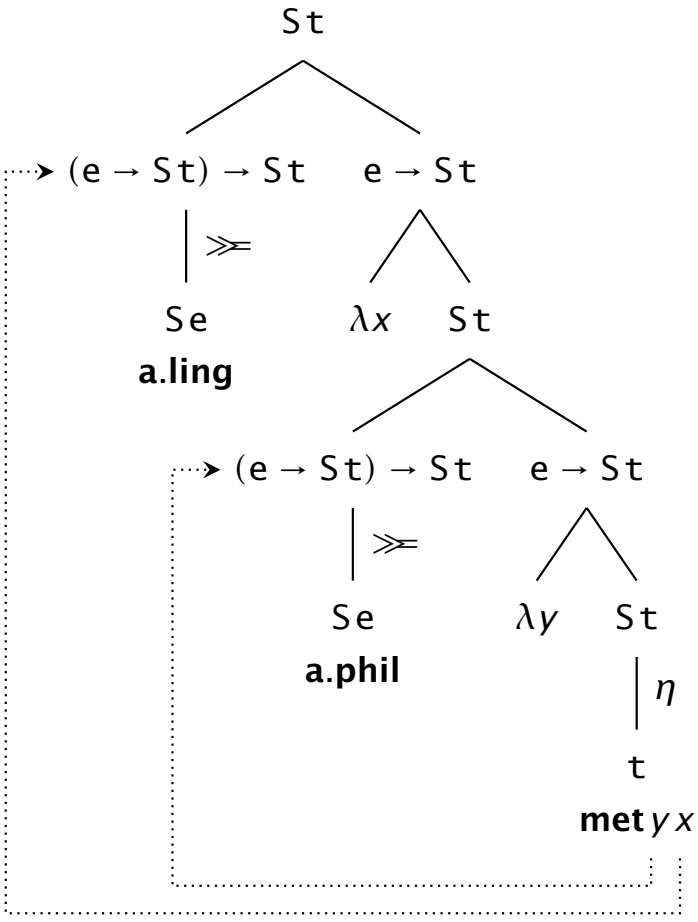
- $\bullet :: \mathsf{S}\,a \to (a \to b) \to \mathsf{S}\,b$
- $\bullet := \lambda m.\lambda f.\{f\,x \mid x \in m\}$

$$\mu :: \mathsf{S}\,(\mathsf{S}\,a) \to \mathsf{S}\,a$$
$$\mu := \lambda M.\bigcup M$$

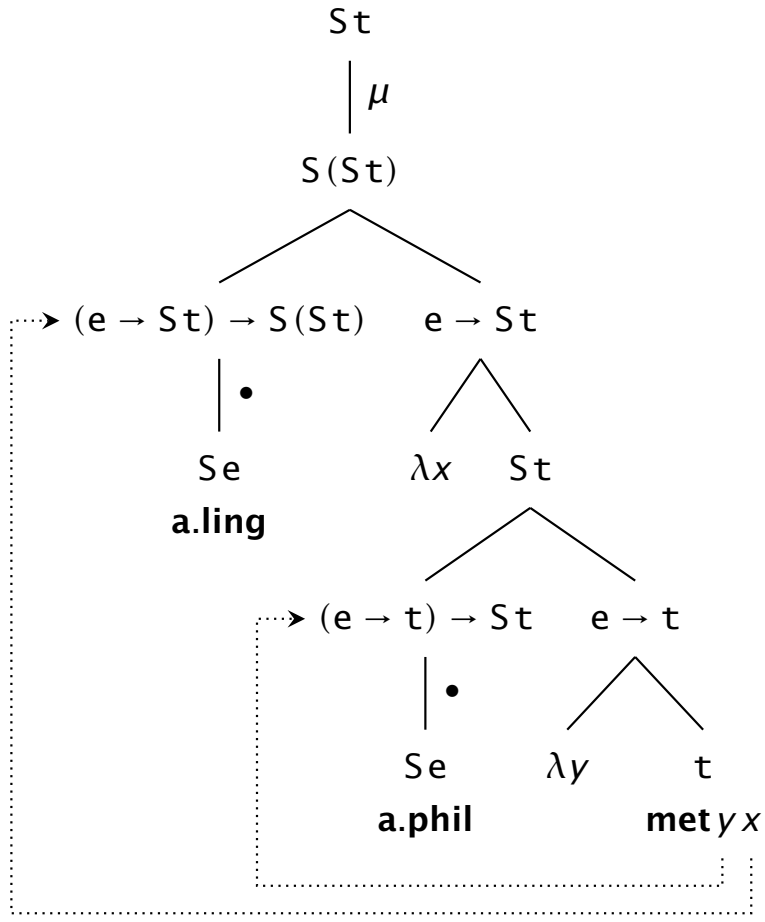
These functions relate to $\gg=$ as follows:

$$\begin{aligned}(m \bullet f)^{\mu} &= \bigcup (m \bullet f) \\ &= \bigcup \{f\,x \mid x \in m\} \\ &= \bigcup_{x \in m} f\,x \\ &= m \gg= f\end{aligned}$$

Comparing different styles



$= \{\textbf{met}\, y\, x \mid \textbf{ling}\, x, \textbf{phil}\, y\}$



$= \{\textbf{met}\, y\, x \mid \textbf{ling}\, x, \textbf{phil}\, y\}$

Higher-order meanings and selectivity

Data: selectivity

Indefinites on an island take scope **in different ways** outside the island:

1. If [a persuasive lawyer visits a relative of mine], I'll inherit a house.

   $\checkmark \exists_{\textbf{lawyer}} \gg \textbf{if} \gg \exists_{\textbf{relative}}$, $\checkmark \exists_{\textbf{relative}} \gg \textbf{if} \gg \exists_{\textbf{lawyer}}$,
   $\checkmark \exists_{\textbf{lawyer}} \gg \exists_{\textbf{relative}} \gg \textbf{if}$

2. Every grad would be overjoyed if [some paper on indefinites was discussed in a popular grad seminar being offered this term].

   $\checkmark \exists_{\textbf{seminar}} \gg \forall \gg \exists_{\textbf{paper}} \gg \textbf{if}$

Indeed, such behavior seems to be essentially presupposed (if not directly argued for) by the dominant accounts of exceptionally scoping indefinites (cf. Reinhart 1997, Brasoveanu & Farkas 2011).

Cross-categorial selectivity

Seen in association with focus (Rooth 1996, Wold 1996) (tho somewhat controversial), as well as binding (where it's more or less presupposed):

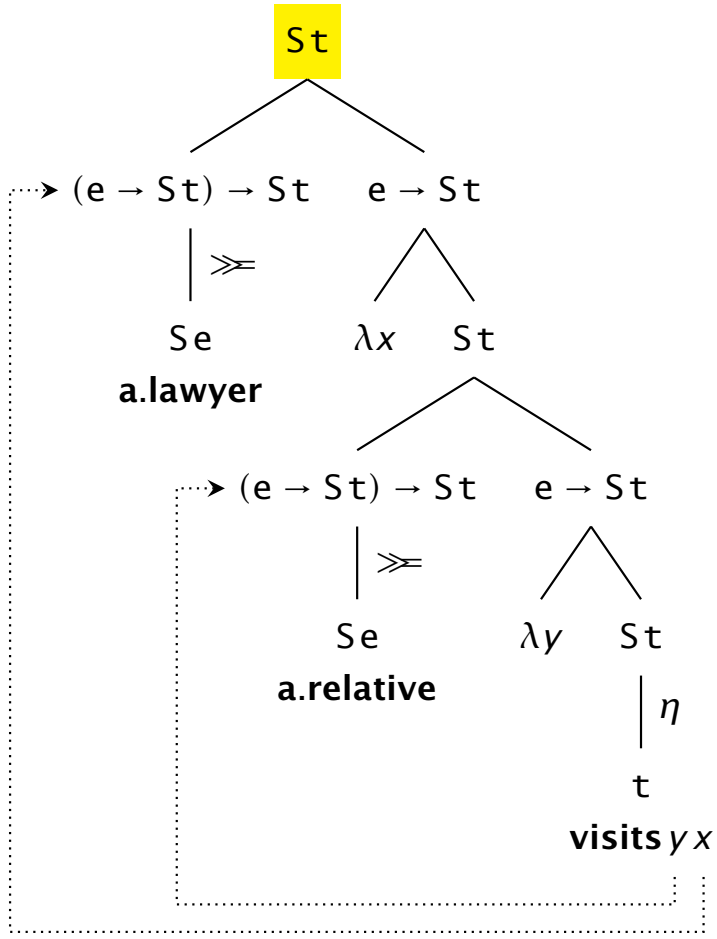
1. [John only gripes when [MARY$_F$ leaves the lights on]]$_C$, and [MARY$_F$ only gripes when [[JOHN$_F$]$_F$ leaves the lights on]]$_{GSmC}$

2. Every boy^x told every girl^y that it'd be great if [she$_y$ cited him$_x$].

Examples in other domains (supplementation, presupposition) are hard to cook up since both have a *strong* preference for maximal projection.

- But you might see if you could construct (e.g.) cases with two presupposition triggers on an island, one of which is locally accommodated (Heim 1983), one of which projects all the way up.

Building the island. . .



$$= \{\textbf{visits}\, y\, x \mid \textbf{lawyer}\, x, \textbf{relative}\, y\}$$

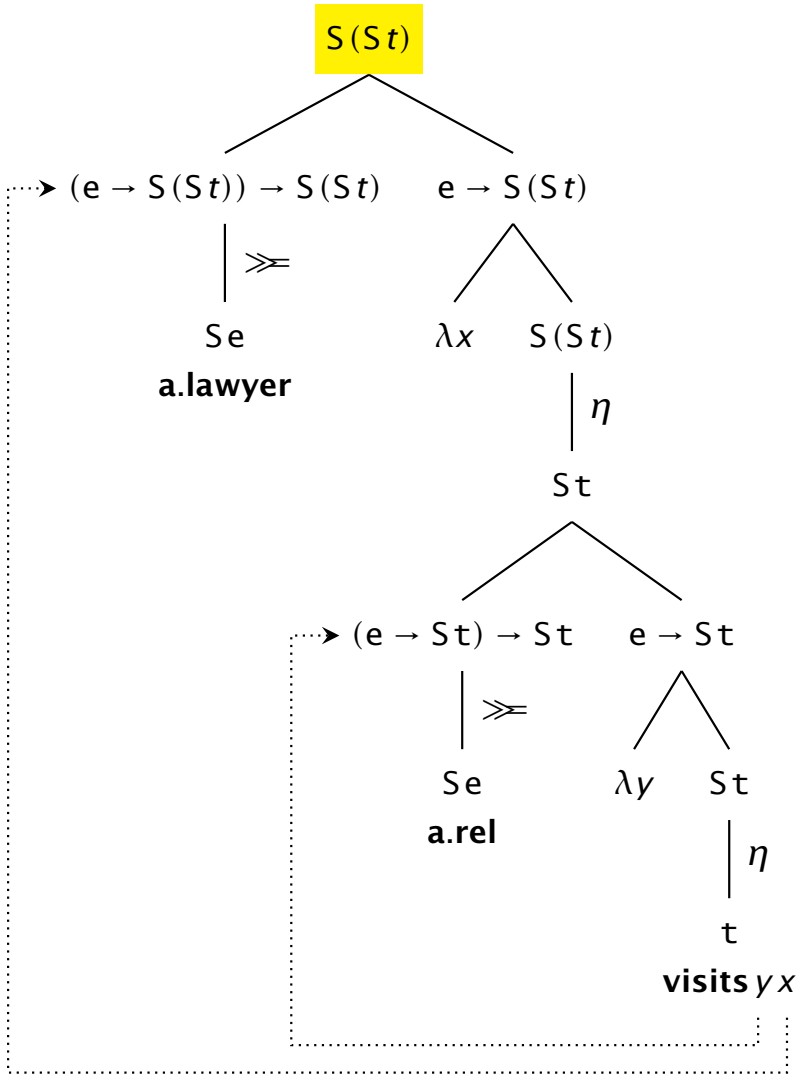
Executing our old exceptional scope trick. . .

$$\{\textbf{visits}\ y\ x \mid \textbf{lawyer}\ x, \textbf{relative}\ y\}^{\gg=} = \lambda f.\bigcup_{p \in \{\textbf{visits}\ y\ x \mid \textbf{lawyer}\ x, \textbf{relative}\ y\}} f\ p$$
$$= \lambda f.\bigcup_{\textbf{lawyer}\ x, \textbf{relative}\ y} f\ (\textbf{visits}\ y\ x)$$

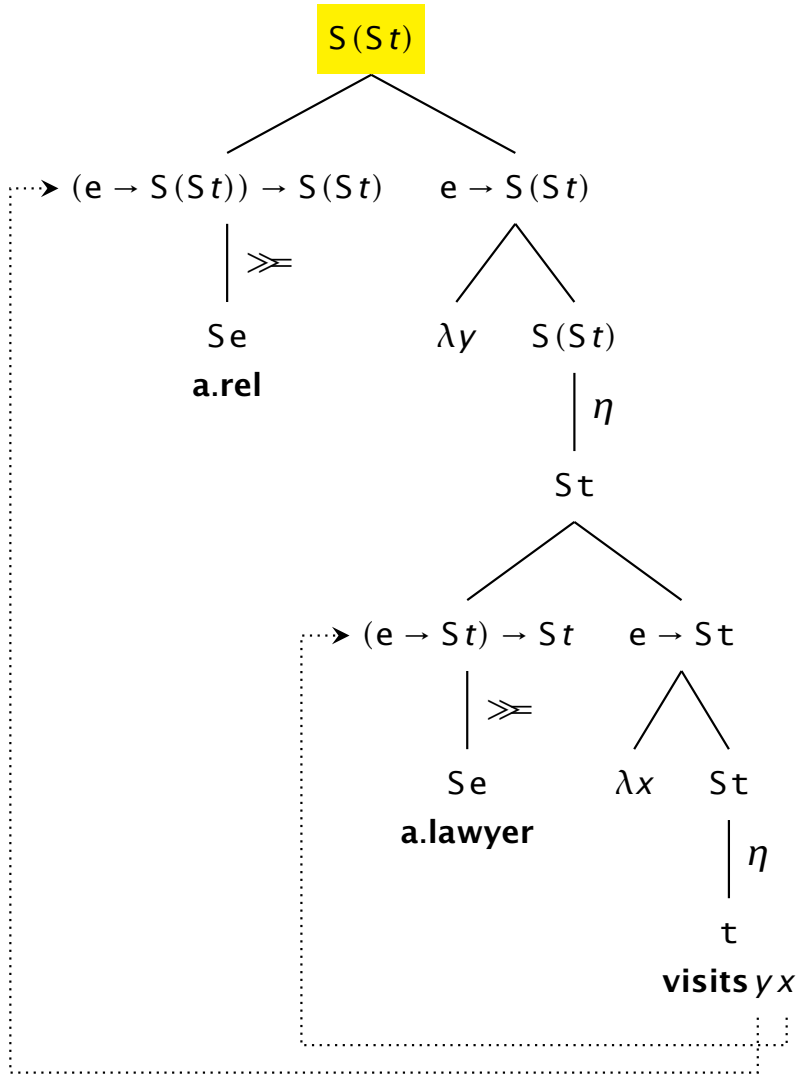
Oops. . . Looks like we've given *both* indefinites scope out of the island.

- Certainly, this is a possible reading (so, no over-generation)
- But it's not the *only* reading (so, under-generation?)

Building higher-order meanings



$= \{\{\textbf{visits}\, y\, x \mid \textbf{rel}\, y\} \mid \textbf{lawyer}\, x\}$



$= \{\{\textbf{visits}\, y\, x \mid \textbf{lawyer}\, x\} \mid \textbf{rel}\, y\}$

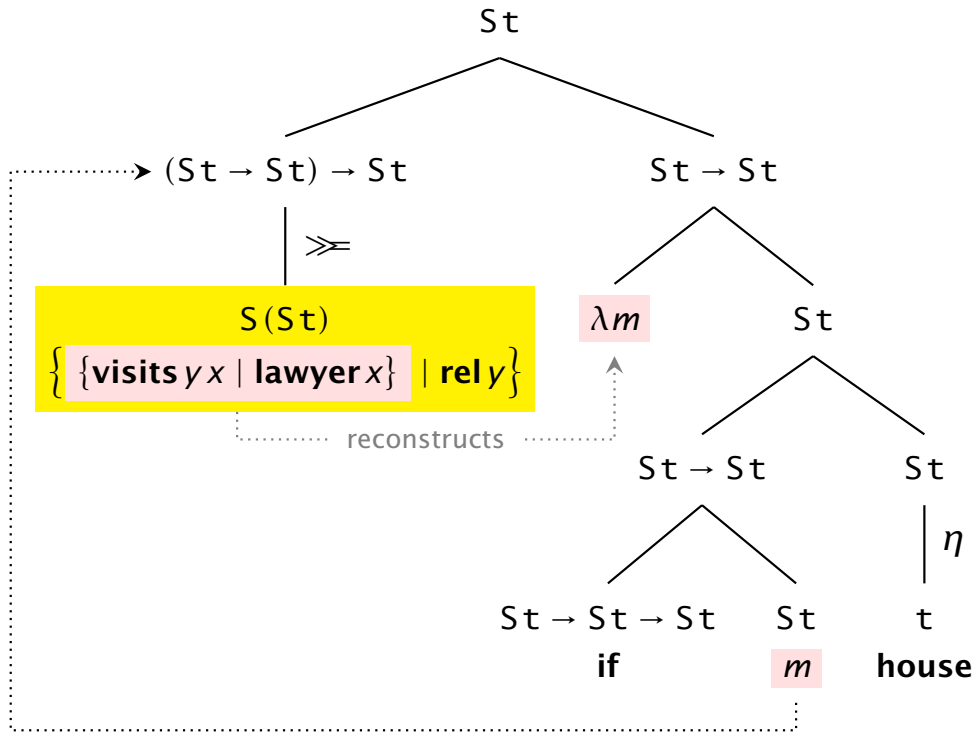
Higher-order meanings

If the lawyers are $\mathbf{L_1}$ and $\mathbf{L_2}$ and the relatives are $\mathbf{R_1}$ and $\mathbf{R_2}$, these higher-order sets amount to the following:

$$\left\{\begin{array}{l}\{\mathbf{visits}\,\mathbf{R_1}\,\mathbf{L_1},\,\mathbf{visits}\,\mathbf{R_2}\,\mathbf{L_1}\},\\ \{\mathbf{visits}\,\mathbf{R_1}\,\mathbf{L_2},\,\mathbf{visits}\,\mathbf{R_2}\,\mathbf{L_2}\}\end{array}\right\} \qquad \left\{\begin{array}{l}\{\mathbf{visits}\,\mathbf{R_1}\,\mathbf{L_1},\,\mathbf{visits}\,\mathbf{R_1}\,\mathbf{L_2}\},\\ \{\mathbf{visits}\,\mathbf{R_2}\,\mathbf{L_1},\,\mathbf{visits}\,\mathbf{R_2}\,\mathbf{L_2}\}\end{array}\right\}$$

Alternatively, in lieu of an extra η, we could have used $\bullet$ to build up these meanings, and foregone the final μ.

An exceptional scope derivation



$$= \{\textbf{if}\,(\exists x \in \textbf{lawyer} : \textbf{visits}\,y\,x)\,\textbf{house} \mid \textbf{rel}\,y\}$$

Generalized selectivity

Imagine you have *n* indefinites on an island (and that none of them contains or binds into any of the others). How many maximally higher-order derivations will be available for the island?

Generalized selectivity

Imagine you have *n* indefinites on an island (and that none of them contains or binds into any of the others). How many maximally higher-order derivations will be available for the island?

$$n!$$

I.e., as many as there are scopal permutations of the indefinites on the island. In other words, we'll always be able to distinguish any number of (independent) indefinites from each other, and in any order.

Summing up

So what this tells us is that we can exert a lot of control over which pieces of the island are evaluated where.

Using higher-order meanings (which come for free!) we can distinguish different layers of indefinite-ness, in a way that allows different indefinites on an island to be distinguished outside the island.

Binding

Basic data

Indefinites interact with pronouns (obvi!):

1. **Indefinites bind pronouns:**
   A candidate^x submitted her$_x$ paper.

2. **Indefinites can be bound into:**
   A candidate^x submitted a paper she$_x$ had written.

3. **...Including by quantifiers:**
   No candidate^x submitted a paper she$_x$ had written.

Binding into an island

A key piece of data:

1. Everybody^{x} loves when [a famous expert on indefinites cites him$_{x}$].
   $\forall \gg {}^{\checkmark}\exists$

The indefinite can take exceptional scope over the matrix subject. On our account, this should require the [island] to scope over the subject.

Is that scoping compatible with the subject binding into the island?

A binding monad?

Suppose we were going to define a ‘monad instance’ for binding. How would that look?

The type constructor is easy to, ahem, construct:

A binding monad?

Suppose we were going to define a ‘monad instance’ for binding. How would that look?

The type constructor is easy to, ahem, construct:

$$\mathsf{G}\,a := g \to a$$

And we already know what η would have to be:

A binding monad?

Suppose we were going to define a ‘monad instance’ for binding. How would that look?

The type constructor is easy to, ahem, construct:

$$\mathsf{G}\,a := g \to a$$

And we already know what η would have to be:

$$\eta\, x := \lambda g.\, x$$

What about $\gg=$?

We know it has to have the type $\mathsf{G}\,a \to (a \to \mathsf{G}\,b) \to \mathsf{G}\,b \ldots$

Ditching G for the full blinding glory of the unabbreviated type:

$$(g \to a) \to (a \to g \to b) \to g \to b$$

Oh. . .

Let me show you an interesting trick. . .

```
Djinn> (>>=) ? (g -> a) -> (a -> g -> b) -> g -> b
```

Oh...

Let me show you an interesting trick...

```
Djinn> (>>=) ? (g -> a) -> (a -> g -> b) -> g -> b
(>>=) a b c = b (a c) c
```

In other words: $m \gg\!= f := \lambda g.\, f\,(m\,g)\,g$.

`Djinn` automatically deduced $\gg\!=$, given its type.

Oh...

Let me show you an interesting trick...

```
Djinn> (>>=) ? (g -> a) -> (a -> g -> b) -> g -> b
(>>=) a b c = b (a c) c
```

In other words: $m \gg\!= f := \lambda g.\, f\,(m\,g)\,g$.

`Djinn` automatically deduced $\gg\!=$, given its type.

°_°

Oh...

Let me show you an interesting trick...

```
Djinn> (>>=) ? (g -> a) -> (a -> g -> b) -> g -> b
(>>=) a b c = b (a c) c
```

In other words: $m \gg\!= f := \lambda g.\, f\,(m\,g)\,g$.

`Djinn` automatically deduced $\gg\!=$, given its type.

°_•

Oh...

Let me show you an interesting trick...

```
Djinn> (>>=) ? (g -> a) -> (a -> g -> b) -> g -> b
(>>=) a b c = b (a c) c
```

In other words: $m \gg\!= f := \lambda g.\, f\,(m\,g)\,g$.

`Djinn` automatically deduced $\gg\!=$, given its type.

•_°

Das it

So that is our monad for binding!

$$\eta :: a \to \mathsf{G}\,a \qquad\qquad \gg\!= :: \mathsf{G}\,a \to (a \to \mathsf{G}\,b) \to \mathsf{G}\,b$$
$$\eta := \lambda x.\,\lambda g.\,x \qquad\qquad \gg\!= := \lambda m.\,\lambda f.\,\lambda g.\,f\,(m\,g)\,g$$

Is it a monad?

Well, the η and $\gg=$ operations have the right shape. So all we need to do is check that it obeys **Left** and **Right identity**, along with **Associativity**.

Let's just check that **Left identity** is satisfied:

$$(\eta\, x) \gg= f =$$

Is it a monad?

Well, the η and $\gg=$ operations have the right shape. So all we need to do is check that it obeys **Left** and **Right identity**, along with **Associativity**.

Let's just check that **Left identity** is satisfied:

$$(\eta\, x) \gg= f = (\lambda g.\, x) \gg= f$$

Is it a monad?

Well, the η and $\gg=$ operations have the right shape. So all we need to do is check that it obeys **Left** and **Right identity**, along with **Associativity**.

Let's just check that **Left identity** is satisfied:

$$\begin{aligned}(\eta\, x) \gg= f &= (\lambda g.\, x) \gg= f \\ &= \lambda h.\, f\, ((\lambda g.\, x)\, h)\, h\end{aligned}$$

Is it a monad?

Well, the η and $\gg=$ operations have the right shape. So all we need to do is check that it obeys **Left** and **Right identity**, along with **Associativity**.

Let's just check that **Left identity** is satisfied:

$$\begin{aligned}(\eta\, x) \gg= f &= (\lambda g.\, x) \gg= f \\ &= \lambda h.\, f\, ((\lambda g.\, x)\, h)\, h \\ &= \lambda h.\, f\, x\, h\end{aligned}$$

Is it a monad?

Well, the η and $\gg=$ operations have the right shape. So all we need to do is check that it obeys **Left** and **Right identity**, along with **Associativity**.

Let's just check that **Left identity** is satisfied:

$$\begin{aligned}(\eta\, x) \gg= f &= (\lambda g.\, x) \gg= f \\ &= \lambda h.\, f\, ((\lambda g.\, x)\, h)\, h \\ &= \lambda h.\, f\, x\, h \\ &= f\, x\end{aligned}$$

Is it a monad?

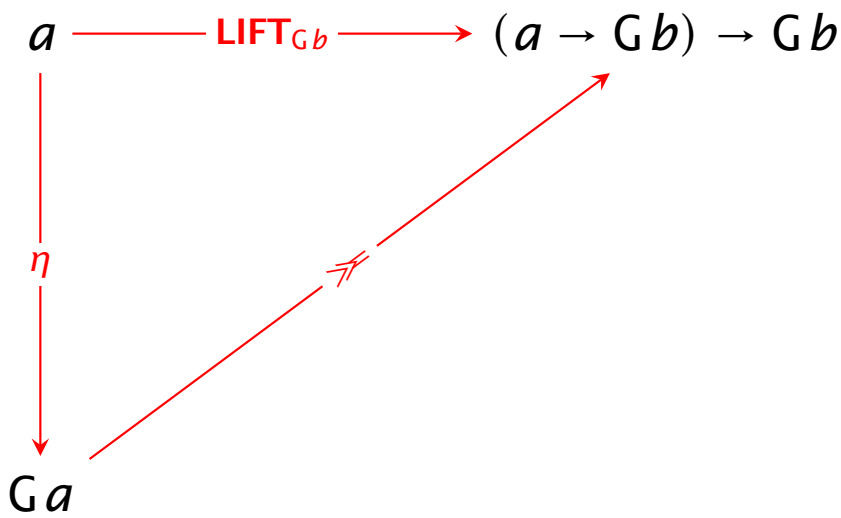
Well, the η and $\gg=$ operations have the right shape. So all we need to do is check that it obeys **Left** and **Right identity**, along with **Associativity**.

Let's just check that **Left identity** is satisfied:

$$\begin{aligned}(\eta\, x) \gg= f &= (\lambda g.\, x) \gg= f \\ &= \lambda h.\, f\, ((\lambda g.\, x)\, h)\, h \\ &= \lambda h.\, f\, x\, h \\ &= f\, x \qquad\qquad \square\end{aligned}$$

I won't subject you to the rest of them here, but it's actually not too difficult to prove them on your own. Exercise: try it!

Another triangle



G is applicative

$$m \gg\!= \lambda f.\, n \gg\!= \lambda x.\, \eta\,(f\,x) =$$

G is applicative

$$m \gg\!= \lambda f.\, n \gg\!= \lambda x.\, \eta\,(f\,x) = \lambda g.\, m\,g\,(n\,g)$$

Does this remind you of anything?

G is applicative

$$m \gg\!= \lambda f.\, n \gg\!= \lambda x.\, \eta\,(f\,x) = \lambda g.\, m\,g\,(n\,g)$$

Does this remind you of anything? This is simply functional application, enriched with assignments!

Like S (and, indeed, any monad), G is applicative: it supports an enriched form of functional application.

G is a functor

$$m \gg\!= \lambda x.\, \eta\, (f\, x) =$$

G is a functor

$$m \gg\!= \lambda x.\, \eta\,(f\,x) = \lambda g.\, f\,(m\,g)$$

Does this remind you of anything?

G is a functor

$$m \gg\!= \lambda x.\, \eta\,(f\,x) = \lambda g.\, f\,(m\,g)$$

Does this remind you of anything? This is the **mapping** operation for G.

Do you notice anything else?

G is a functor

$$m \gg\!= \lambda x.\eta\,(f\,x) = \lambda g.f\,(m\,g)$$

Does this remind you of anything? This is the **mapping** operation for G. Do you notice anything else? It's **function composition**!

Like S (and, indeed, any monad), G is a functor: it supports a mapping operation (if you're familiar with Jacobson's (1999) treatment of pronouns, this might strike you as familiar; hold that thought!).

G's μ

Can you construct the $\mu :: \mathsf{G}\,(\mathsf{G}\,a) \to \mathsf{G}\,a$ operation for G?

G's μ

Can you construct the $\mu :: \mathsf{G}\,(\mathsf{G}\,a) \to \mathsf{G}\,a$ operation for G?

$$M^{\mu} := \lambda g.\, M\, g\, g$$

Layering the two

A tale of 2 monads

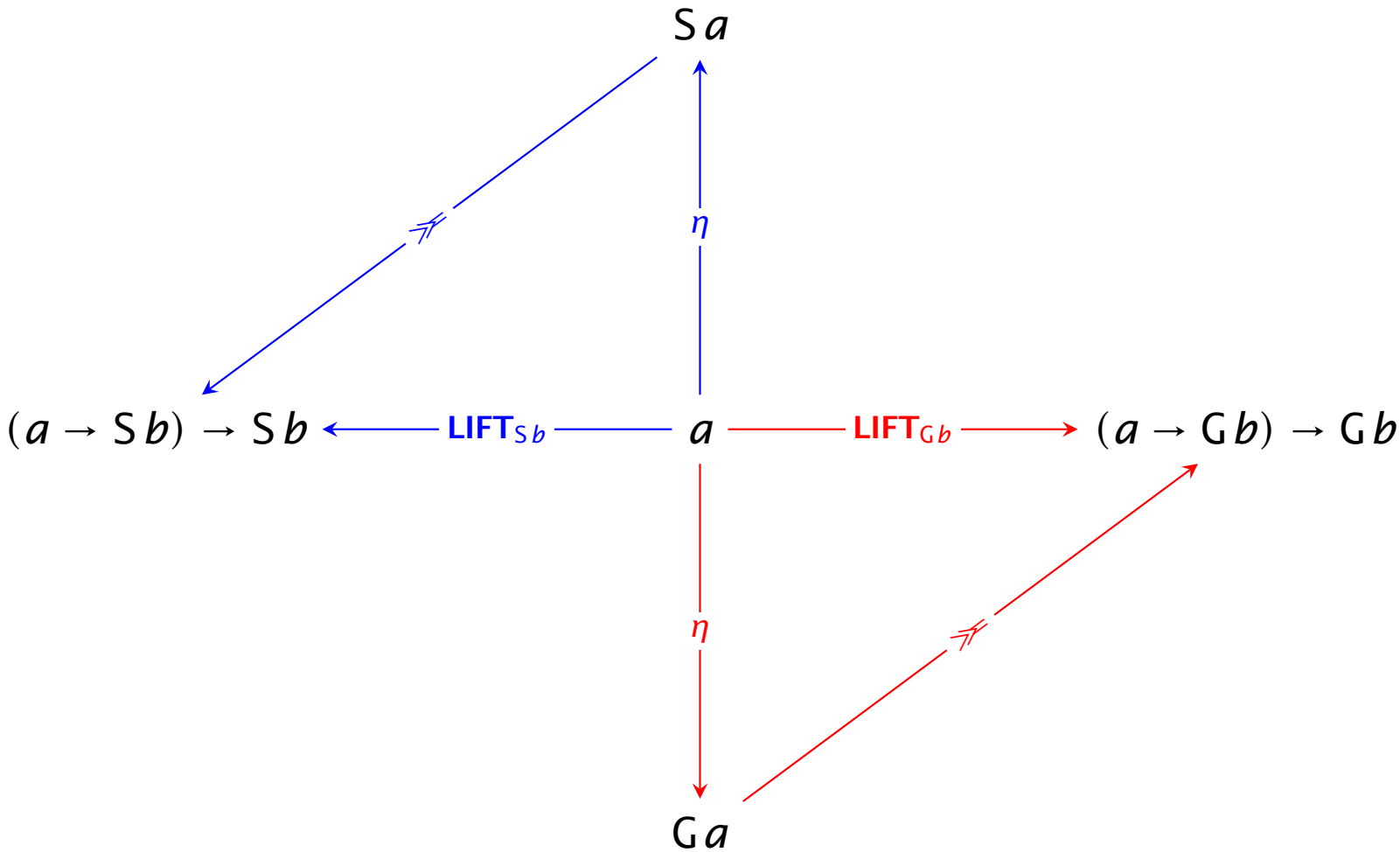
So we have two monads, one for alternatives. . . .

$$\eta :: a \to \mathsf{S}\,a \qquad \gg\!= :: \mathsf{S}\,a \to (a \to \mathsf{S}\,b) \to \mathsf{S}\,b$$
$$\eta := \lambda p.\,\{p\} \qquad \gg\!= := \lambda m.\,\lambda f.\,\bigcup_{x \in m} f\,x$$

. . . And one for binding:

$$\eta :: a \to \mathsf{G}\,a \qquad \gg\!= :: \mathsf{G}\,a \to (a \to \mathsf{G}\,b) \to \mathsf{G}\,b$$
$$\eta := \lambda x.\,\lambda g.\,x \qquad \gg\!= := \lambda m.\,\lambda f.\,\lambda g.\,f\,(m\,g)\,g$$

How they [don't] relate...



A choice

We might wish to derive a *combined* monad so that we might walk and chew gum at the same time.

But as with alternative semantics, this presents us with a choice. Do we layer binding around alternatives, or alternatives around binding?

$$\mathsf{GS}\,a ::= g \to \mathsf{S}\,a \qquad \mathsf{SG}\,a := \mathsf{S}\,(g \to a)$$

Composition for η

For either layering, it's straightforward to derive the corresponding η:

$$\eta_{\mathsf{GS}}\, x :=$$

Composition for η

For either layering, it's straightforward to derive the corresponding η:

$$\eta_{\mathsf{GS}}\, x := \lambda g.\, \{x\}$$

Composition for η

For either layering, it's straightforward to derive the corresponding η:

$$\eta_{GS}\, x := \lambda g.\, \{x\} \qquad\qquad \eta_{SG}\, x :=$$

Composition for η

For either layering, it's straightforward to derive the corresponding η:

$$\eta_{GS}\, x := \lambda g.\, \{x\} \qquad\qquad \eta_{SG}\, x := \{\lambda g.\, x\}$$

You'll notice that each of these operations is a *composition* of η_S and η_G (in different orders):

$$\eta_{GS} \equiv \eta_G \circ \eta_S \qquad\qquad \eta_{SG} \equiv \eta_S \circ \eta_G$$

So *this* much, at least, is automatic. Whichever option we choose, we'll have a ready and waiting η operation.

Applicatives and functors

GS and SG are both applicative:

$$\lambda g.\{f\,x \mid f \in m\,g, x \in n\,g\} \qquad \{\lambda g.f\,g\,(x\,g) \mid f \in m, x \in n\}$$

GS and SG are both functors:

$$\lambda g.\{f\,x \mid x \in m\,g\} \qquad \{\lambda g.f\,(x\,g) \mid x \in m\}$$

But for bind?

It turns out (I think!) that only GS supports a $\gg=$ operation:

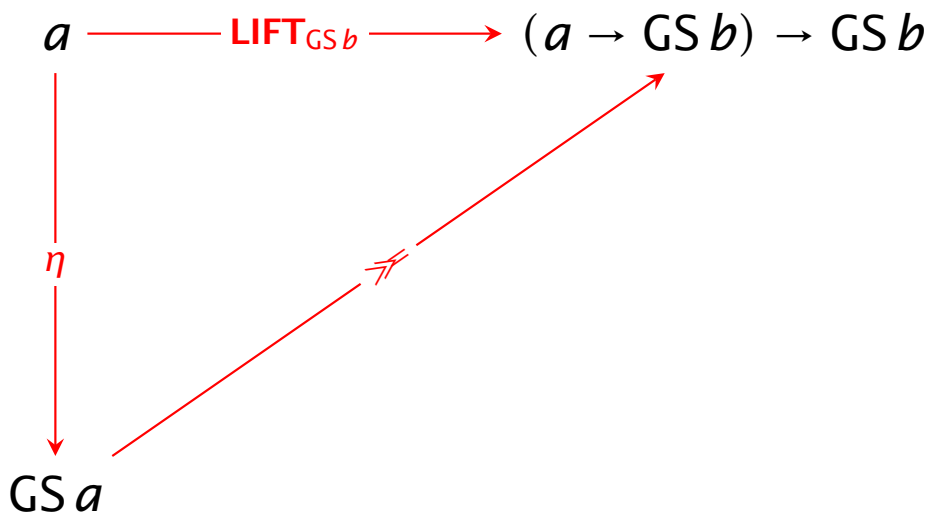
$$m \gg= f := \lambda g. \bigcup_{x \in mg} f\, x\, g$$

Compared to our previous operation, this is just adding some g's:

$$m \gg= f := \bigcup_{x \in m} f\, x$$

So only (I think!) one layering works.

Yet another triangle



Rounding out the picture

Meanings for indefinites:

Rounding out the picture

Meanings for indefinites:

$$\textbf{a.ling} := \lambda g.\{x \mid \textbf{ling}\, x\}$$

Meanings for pronouns:

Rounding out the picture

Meanings for indefinites:

$$\textbf{a.ling} := \lambda g.\{x \mid \textbf{ling}\, x\}$$

Meanings for pronouns:

$$\textbf{she}_0 := \lambda g.\{g_0\}$$

An operator for effecting binding:

Rounding out the picture

Meanings for indefinites:

$$\textbf{a.ling} := \lambda g.\{x \mid \textbf{ling}\, x\}$$

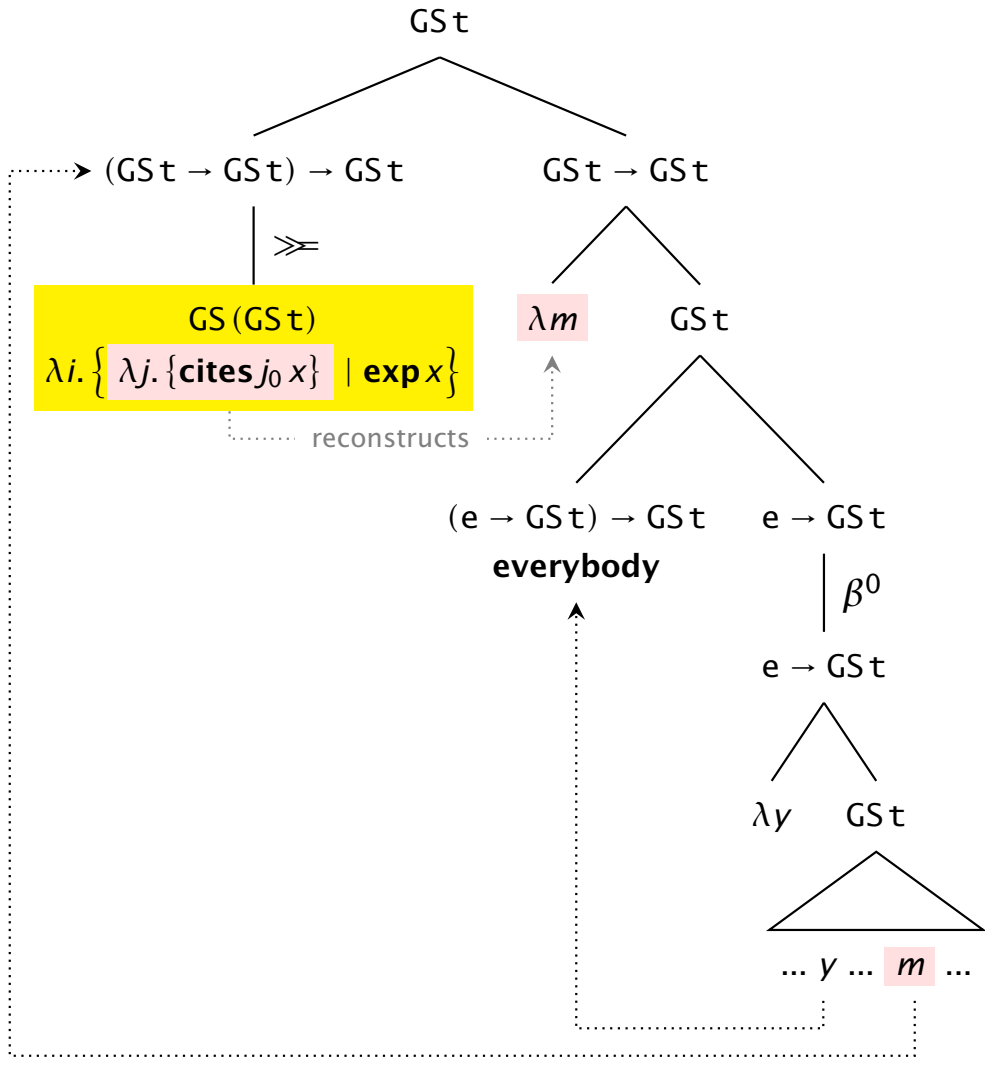
Meanings for pronouns:

$$\textbf{she}_0 := \lambda g.\{g_0\}$$

An operator for effecting binding:

$$\beta^n f := \lambda x.\lambda g.f\, x\, g^{n \to x}$$

Higher-order derivations



Brasoveanu, Adrian & Donka F. Farkas. 2011. How indefinites choose their scope. *Linguistics and Philosophy* 34(1). 1–55. http://dx.doi.org/10.1007/s10988-011-9092-7.

Heim, Irene. 1983. On the projection problem for presuppositions. In Michael Barlow, Daniel P. Flickinger & Michael T. Wescoat (eds.), *Proceedings of the Second West Coast Conference on Formal Linguistics*, 114–125. Stanford: Stanford University Press.

Jacobson, Pauline. 1999. Towards a variable-free semantics. *Linguistics and Philosophy* 22(2). 117–184. http://dx.doi.org/10.1023/A:1005464228727.

Partee, Barbara H. 1986. Noun phrase interpretation and type-shifting principles. In Jeroen Groenendijk, Dick de Jongh & Martin Stokhof (eds.), *Studies in Discourse Representation Theory and the Theory of Generalized Quantifiers*, 115–143. Dordrecht: Foris.

Reinhart, Tanya. 1997. Quantifier scope: How labor is divided between QR and choice functions. *Linguistics and Philosophy* 20(4). 335–397. http://dx.doi.org/10.1023/A:1005349801431.

Rooth, Mats. 1996. Focus. In Shalom Lappin (ed.), *The Handbook of Contemporary Semantic Theory*, 271–298. Oxford: Blackwell.

Wold, Dag E. 1996. Long distance selective binding: The case of focus. In Teresa Galloway & Justin Spence (eds.), *Proceedings of Semantics and Linguistic Theory 6*, 311–328. Ithaca, NY: Cornell University. http://dx.doi.org/10.3765/salt.v6i0.2766.