Focus, monadically and otherwise

Computational semantics seminar

April 5, 2017

Monadic focus

The Roothian picture

Meanings for unfocused and focused expressions:

$$\begin{aligned} [\![\text{Bill}]\!] &:= (\mathbf{b}, \{\mathbf{b}\}) \\ [\![\text{SUE}_F]\!] &:= (\mathbf{s}, \mathbf{alt_s}) \end{aligned}$$

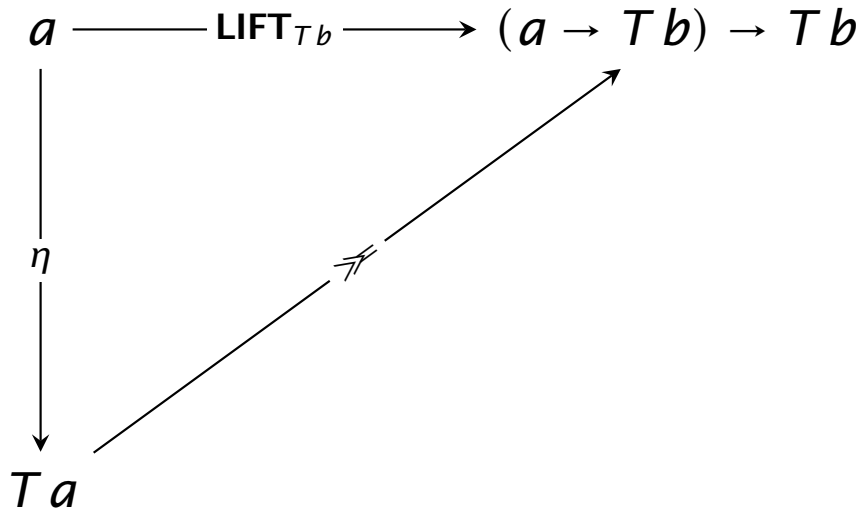
Meaning for binary-branching nodes:

$$[\![X\ Y]\!] := (\underbrace{\mathbf{fst}[\![X]\!](\mathbf{fst}[\![Y]\!])}_{\text{normal meaning}}, \overbrace{\{x\,y \mid x \in \mathbf{snd}[\![X]\!], y \in \mathbf{snd}[\![Y]\!]\}}^{\text{focus meaning}})$$

Normal in the 1st dimension, point-wise in the 2nd.

T _____ T

A **monad** is a type constructor T associated with η and $\gg=$ operations, that makes the following diagram *commute* and satisfies two further laws.



$$m \gg= \eta = m \qquad (\textit{Right Id})$$

$$(m \gg= f) \gg= g = m \gg= \lambda x.\, f\, x \gg= g \qquad (\textit{Assoc})$$

The Focus monad

The **Focus monad** characterizes *regular* composition in the 1st dimension, and *point-wise* (/nondeterministic) composition in the 2nd:

$$
\begin{array}{ll}
\mathsf{F}\, a & ::= \\
\eta\, x & := \\
(x, Y) \gg\!= f & :=
\end{array}
$$

The Focus monad

The **Focus monad** characterizes *regular* composition in the 1st dimension, and *point-wise* (/nondeterministic) composition in the 2nd:

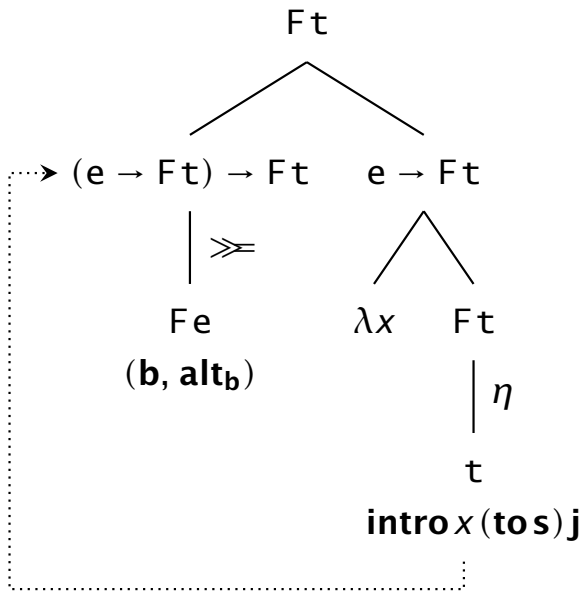
$$
\begin{aligned}
\mathsf{F}\, a &::= (a, \mathsf{S}\, a) \\
\eta\, x &:= (x, \{x\}) \\
(x, Y) \gg\!= f &:= \left(\mathbf{fst}\,(f\, x), \bigcup_{y \in Y} \mathbf{snd}\,(f\, y)\right)
\end{aligned}
$$

F-marking

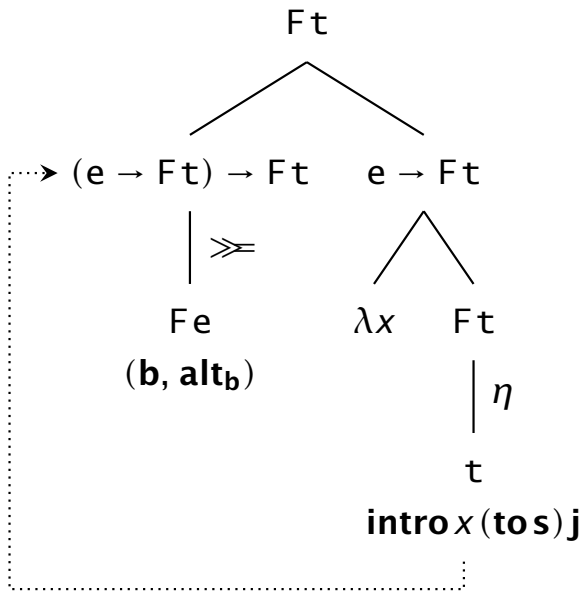
$$\cdot_{\mathsf{F}} :: a \to \mathsf{F}\, a$$
$$\cdot_{\mathsf{F}} := \lambda x.\, (x, \mathbf{alt}_x)$$

[Same type as η, but a different function.]

Basic derivation



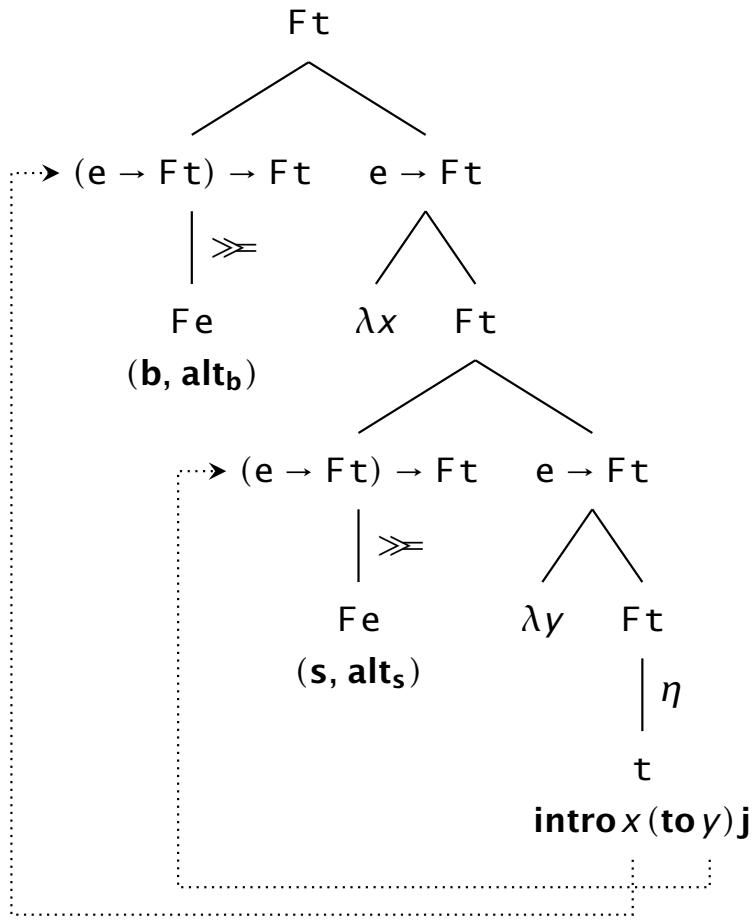
Basic derivation



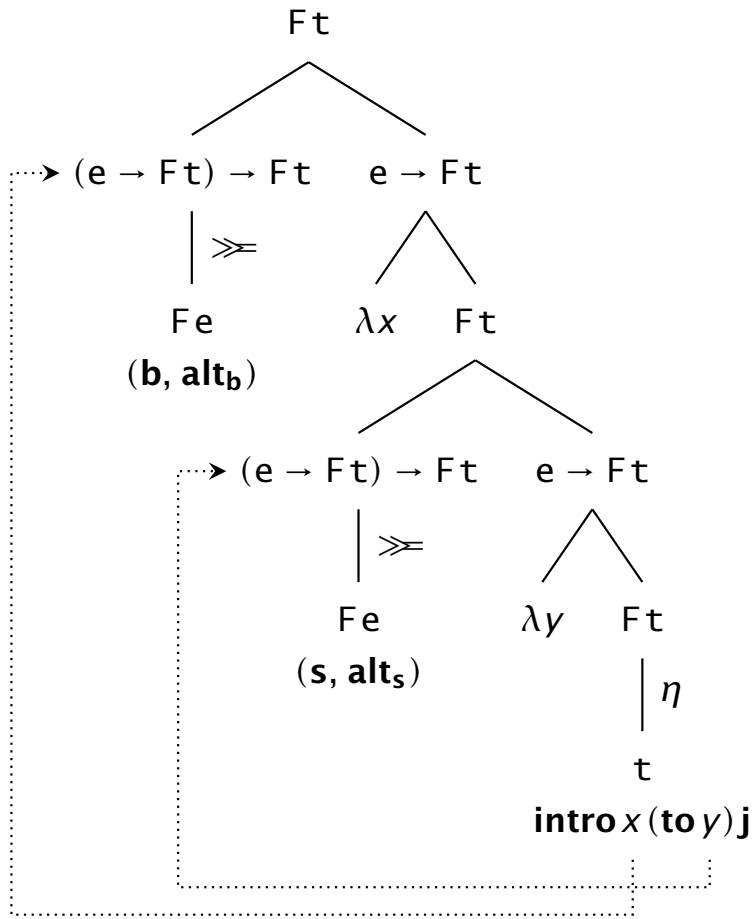
$$(\mathbf{intro}\,\mathbf{b}\,(\mathbf{to}\,\mathbf{s})\,\mathbf{j},\ \{\mathbf{intro}\,x\,(\mathbf{to}\,\mathbf{s})\,\mathbf{j} \mid x \in \mathbf{alt_b}\})$$

No need to specify $\{\!\{\cdot\}\!\}$ for each terminal node, or say how $\{\!\{\cdot\}\!\}$ composes two meanings. Just functional application, and **scope**.

Multiple foci

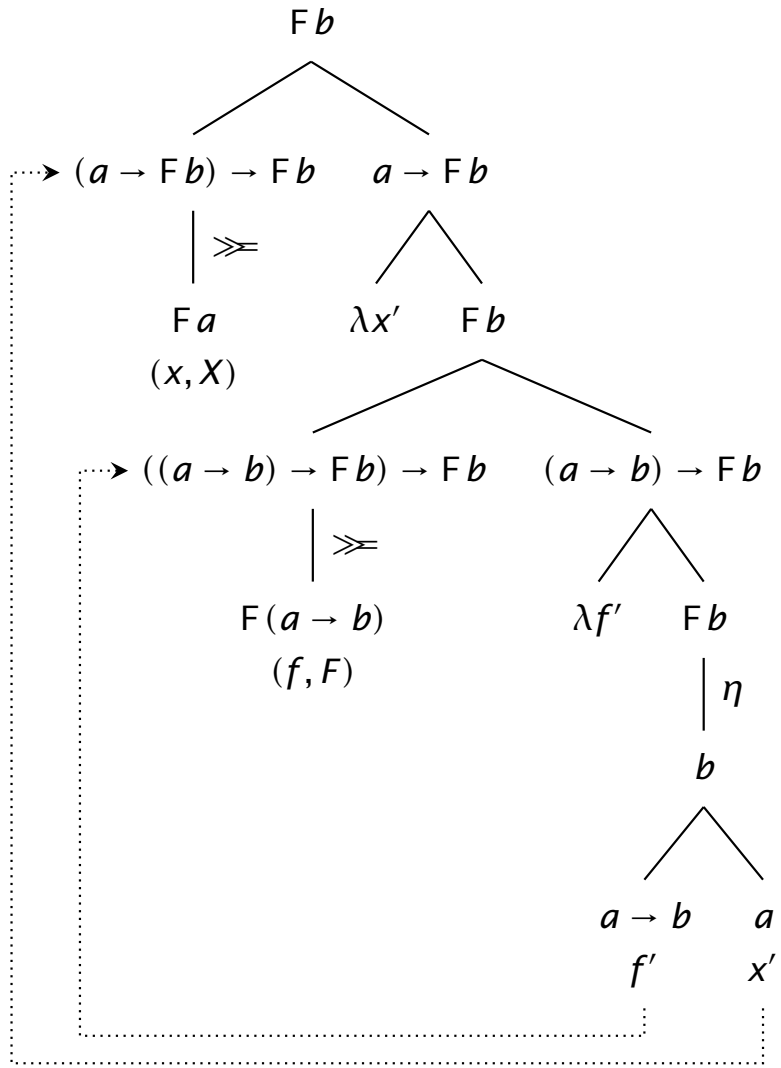


Multiple foci

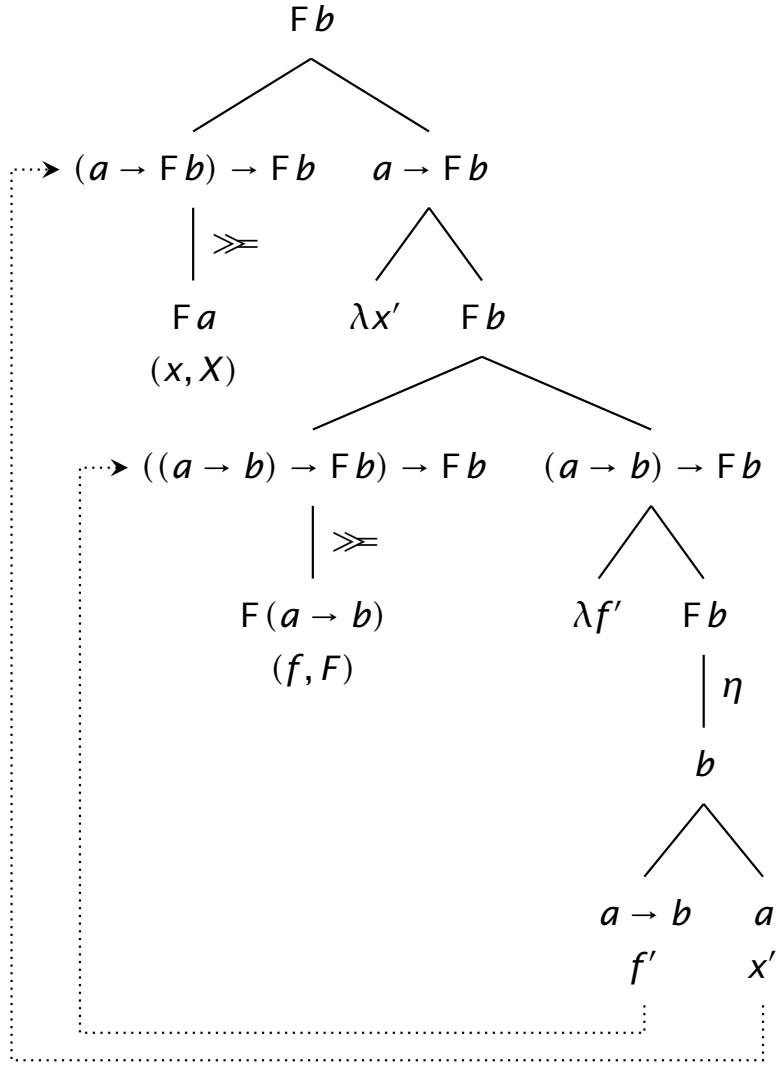


$$(\mathbf{intro}\,\mathbf{b}\,(\mathbf{to}\,\mathbf{s})\,\mathbf{j},\ \{\mathbf{intro}\,x\,(\mathbf{to}\,y)\,\mathbf{j} \mid x \in \mathbf{alt_b}, y \in \mathbf{alt_s}\})$$

Monadic application



Monadic application



$$(f\,x, \{f'\,x' \mid x' \in X, f' \in F\})$$

Association with focus

A type and meaning for (propositional) *only*:

$$\begin{aligned} &\textbf{only} && :: \mathsf{F}\,\mathtt{t} \to \mathtt{t} \\ &\textbf{only}\,(p, A) && := \lambda w.\, p\,w \wedge \forall\, q \in A : q\,w \Rightarrow q = p \end{aligned}$$

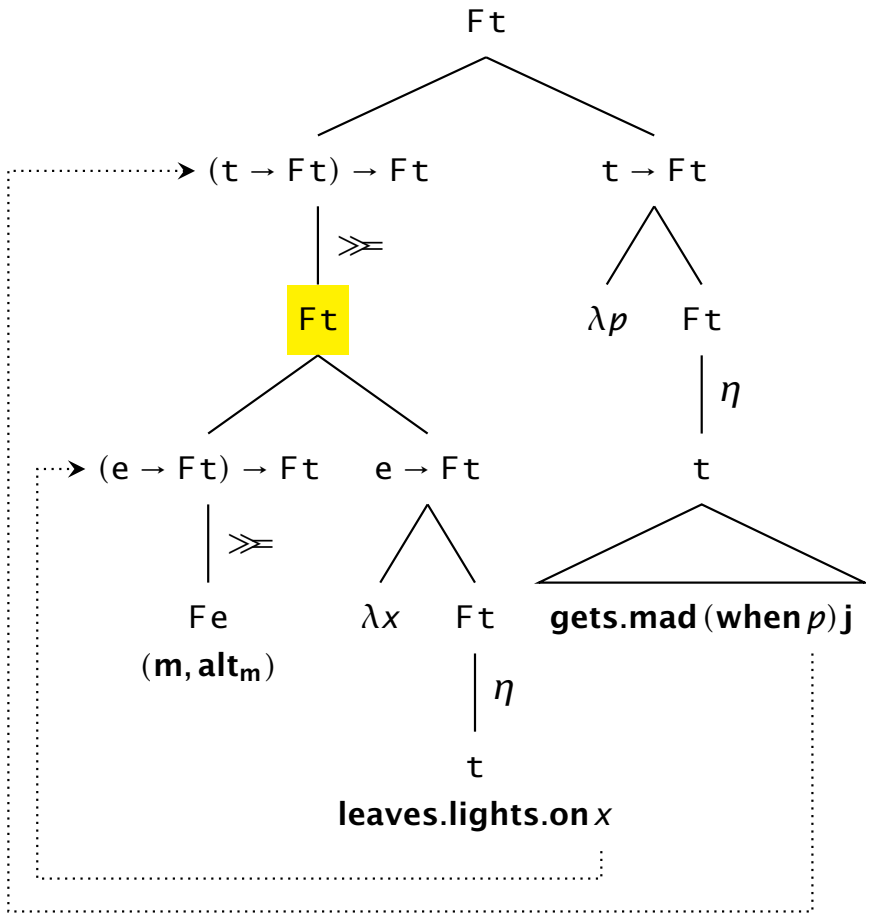
A type and meaning for the focus interpretation operator:

$$\cdot \sim C :: \mathsf{F}\,a \to a$$

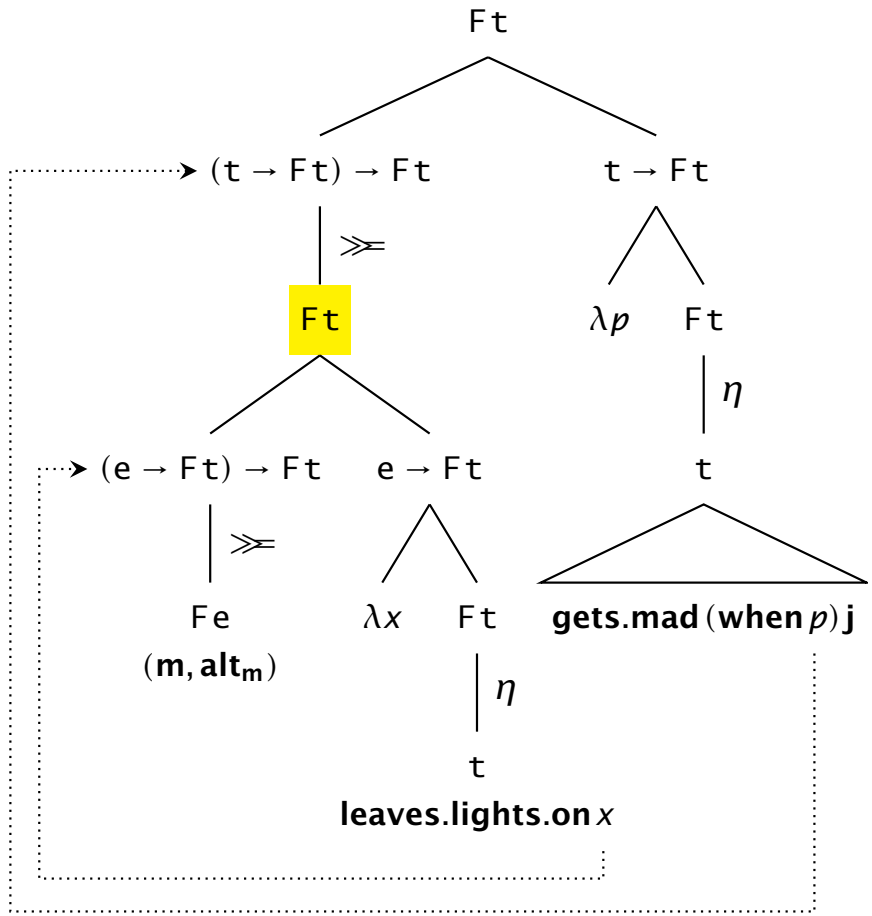
$$(x, A) \sim C := \begin{cases} x \text{ if } A \sim C \\ \# \text{ otherwise} \end{cases}$$

Both provided *categorematically*.

Islands


Ft
(t → Ft) → Ft
t → Ft
≫=
Ft
λp
Ft
η
(e → Ft) → Ft
e → Ft
t
≫=
Fe
(m, alt_m)
λx
Ft
gets.mad (when p) j
η
t
leaves.lights.on x

Islands



$$(\mathbf{mad}\,(\mathbf{when}\,(\mathbf{llo}\,\mathbf{m}))\,\mathbf{j},\ \{\mathbf{mad}\,(\mathbf{when}\,(\mathbf{llo}\,x))\,\mathbf{j} \mid x \in \mathbf{alt_m}\})$$

Selectivity

As ever, higher-order meanings for focused expressions allow us to secure selectivity outside scope islands:

$$m \gg\!= \lambda x.\, \eta\, (n \gg\!= \lambda y.\, \eta\, (\ldots x \ldots y \ldots)) :: \mathrm{F}\,(\mathrm{F}\,a)$$

(cf. the 'iterated C' approach to higher-order questions)

An example of a higher-order focus meaning (compare to higher-order alternative sets used for questions/indefinites):

$$\Big((\mathbf{saw}\,\mathbf{m}\,\mathbf{j}, \{\mathbf{saw}\,x\,\mathbf{j} \mid x \in \mathbf{alt_m}\}), \Big\{ (\mathbf{saw}\,\mathbf{m}\,y, \{\mathbf{saw}\,x\,y \mid x \in \mathbf{alt_m}\}) \mid y \in \mathbf{alt_j} \Big\} \Big)$$

Selectivity

As ever, higher-order meanings for focused expressions allow us to secure selectivity outside scope islands:

$$m \gg\!= \lambda x.\, \eta\, (n \gg\!= \lambda y.\, \eta\, (\ldots x \ldots y \ldots)) :: \mathrm{F}\,(\mathrm{F}\,a)$$

(cf. the 'iterated C' approach to higher-order questions)

An example of a higher-order focus meaning (compare to higher-order alternative sets used for questions/indefinites):

$$\Big((\mathbf{saw\,m\,j}, \{\mathbf{saw}\,x\,\mathbf{j} \mid x \in \mathbf{alt_m}\}), \Big\{ (\mathbf{saw\,m}\,y, \{\mathbf{saw}\,x\,y \mid x \in \mathbf{alt_m}\}) \mid y \in \mathbf{alt_j} \Big\} \Big)$$

OMG.

For selectivity

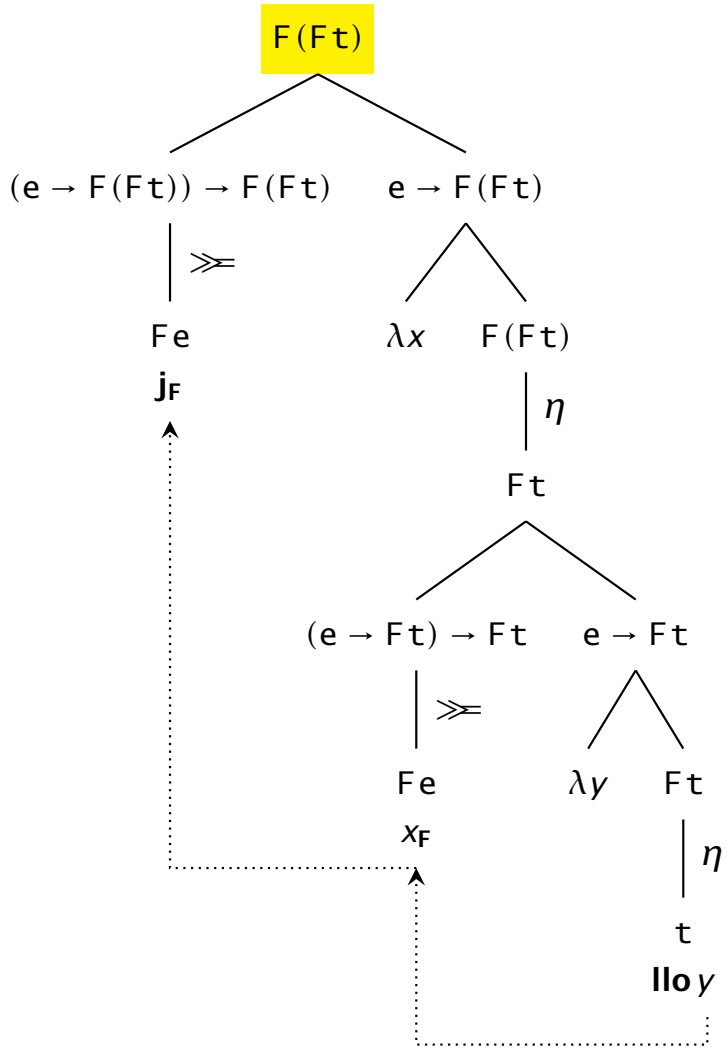
Association with focus alternatives is selective outside islands:

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]]$_{\sim C}$

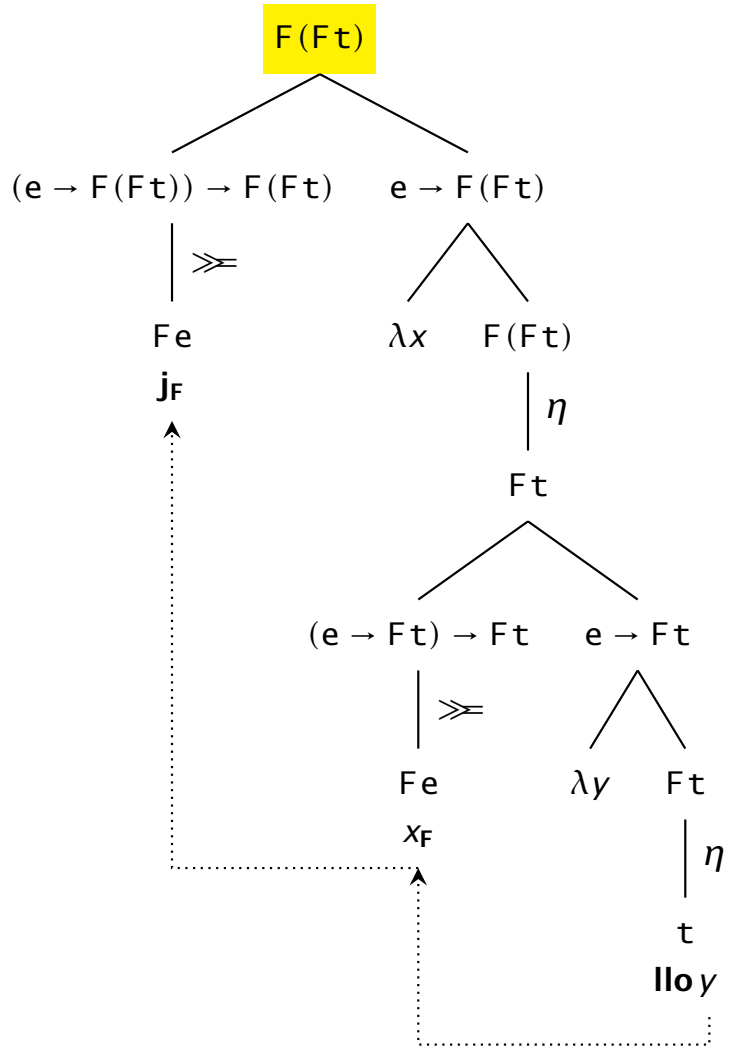
JOHN needs to associate both with *only* and with $\sim$. But because *JOHN* is embedded in an island, all of its F-marks are swallowed up by *only*!

(This is also problematic for structured meanings.)

A higher-order meaning for the island

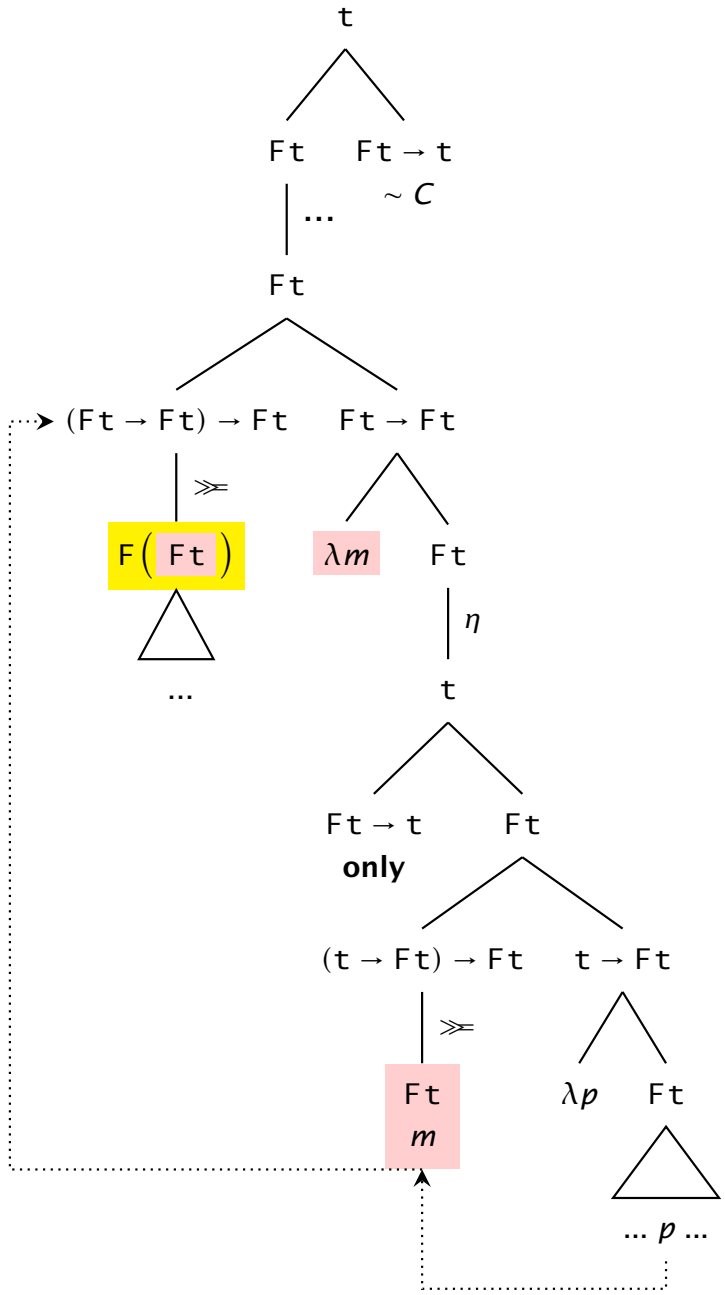


A higher-order meaning for the island



$$\Big(\big(\mathbf{llo}\,\mathbf{j}, \{\mathbf{llo}\,y \mid y \in \mathbf{alt_j}\}\big), \Big\{\big(\mathbf{llo}\,x, \{\mathbf{llo}\,y \mid y \in \mathbf{alt}_x\}\big) \mid x \in \mathbf{alt_j}\Big\}\Big)$$

Deriving selectivity



Interaction

Binding

We've encountered a number of possible treatments of binding:

- Static: $\mathsf{G}\,a ::= \mathsf{g} \to a$ (the **Reader** monad)
- Dynamic: $\mathsf{H}\,a ::= \mathsf{g} \to (a, \mathsf{g})$ (the **State** monad)
- Static + alternatives: $\mathsf{GS}\,a ::= \mathsf{g} \to \mathsf{S}\,a$ (the **Reader-Set** monad)
- Dynamic + alternatives: $\mathsf{D}\,a ::= \mathsf{g} \to \mathsf{S}\,(a, \mathsf{g})$ (the **State-Set** monad)

Let's consider how the Focus monad might interact with these. We'll confine our attention to the simplest monad, G.

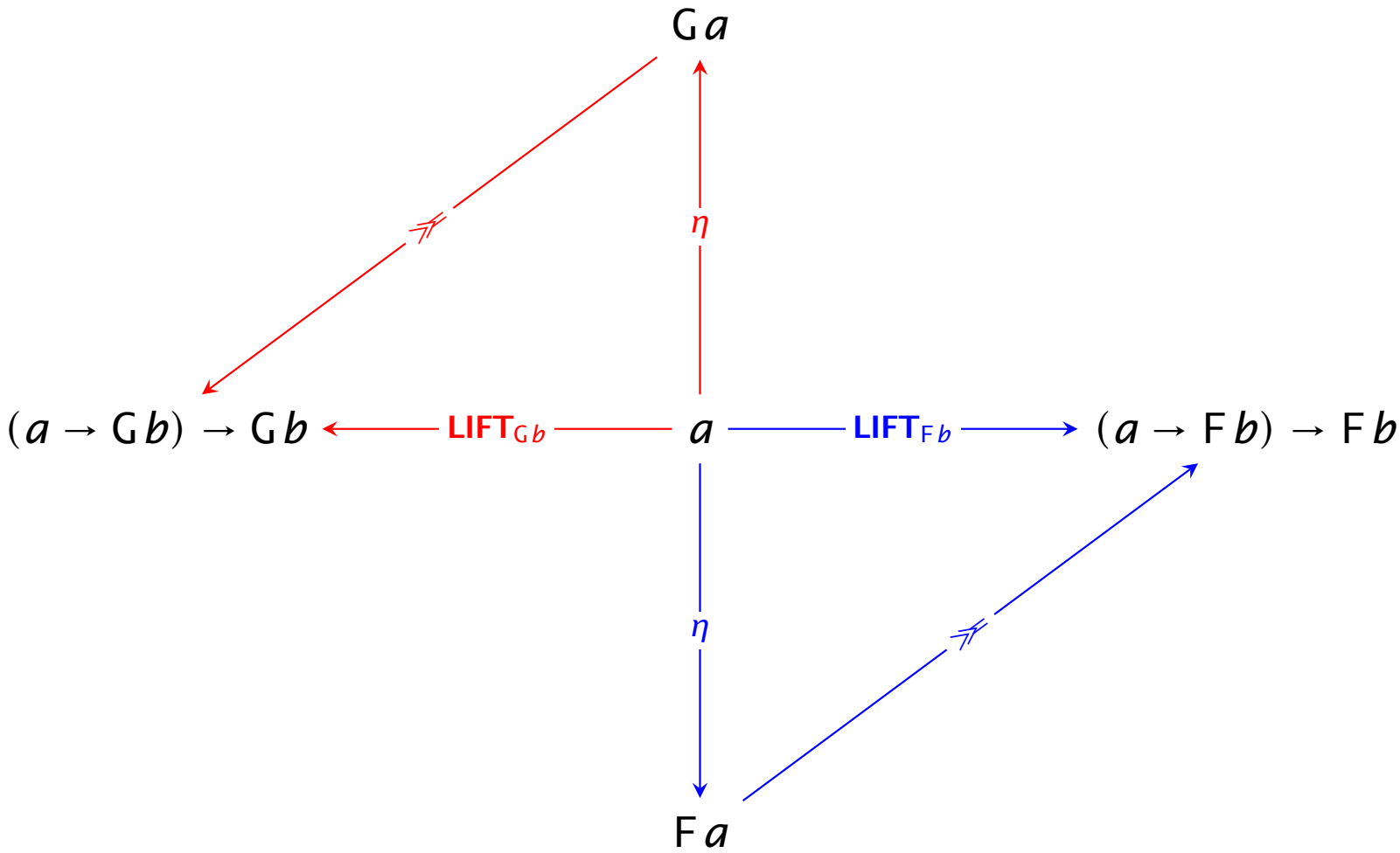
G reminder

$$
\begin{aligned}
\mathsf{G}\,a \quad & ::= \\
\eta\,x \quad & := \\
m \gg\!= f \quad & :=
\end{aligned}
$$

G reminder

$$
\begin{aligned}
\mathsf{G}\,a \quad & ::= \mathsf{g} \to a \\
\eta\, x \quad & := \lambda g.\, x \\
m \gg\!= f \; & := \lambda g.\, f\,(m\,g)\,g
\end{aligned}
$$

How they [don't] relate...



Two possibilities for layering

As with supplements, we can layer these monads on top of each other, without any need to directly define a combined monad:

Assignment-dependent focused meanings: $\mathsf{G}\,(\mathsf{F}\,a)$
Focused assignment-dependent meanings: $\mathsf{F}\,(\mathsf{G}\,a)$

Generating both of these orderings:

$$m \gg\!= \lambda x.\, \eta\,(n \gg\!= \lambda y.\, \eta\,(\ldots\, x\, \ldots\, y\, \ldots)) :: \mathsf{G}\,(\mathsf{F}\,a)$$
$$m \gg\!= \lambda x.\, \eta\,(n \gg\!= \lambda y.\, \eta\,(\ldots\, x\, \ldots\, y\, \ldots)) :: \mathsf{F}\,(\mathsf{G}\,a)$$

Assignments over focus alternatives

Here's a sample meaning of type $\mathsf{G}(\mathsf{F}\,\mathtt{t})$:

$$\lambda g.\,(\mathbf{saw}\,g_0\,\mathbf{m},\,\{\mathbf{saw}\,g_0\,x \mid x \in \mathbf{alt_m}\})$$

Here's another possibility, type $\mathsf{G}(\mathsf{F}\,\mathtt{e})$ for a *focused pronoun*:

$$\lambda g.\,(g_0,\,\mathbf{alt}_{g_0})$$

Focus alternatives over assignments

Here's a sample meaning of type $\mathsf{F}(\mathsf{G}\,\mathtt{t})$:

$$(\lambda g.\,\mathbf{saw}\,g_0\,\mathbf{m},\,\{\lambda g.\,\mathbf{saw}\,g_0\,x \mid x \in \mathbf{alt_m}\})$$

Here's another possibility, type $\mathsf{F}(\mathsf{G}\,\mathtt{e})$ for a *focused pronoun*:

$$\left(\lambda g.\,g_0,\,\mathbf{alt}_{\lambda g.\,g_0}\right)$$

Deriving both layerings

Alternatives over assignments is super easy to derive:

$$\left(\lambda g.\, g_0, \mathbf{alt}_{\lambda g.\, g_0}\right) =$$

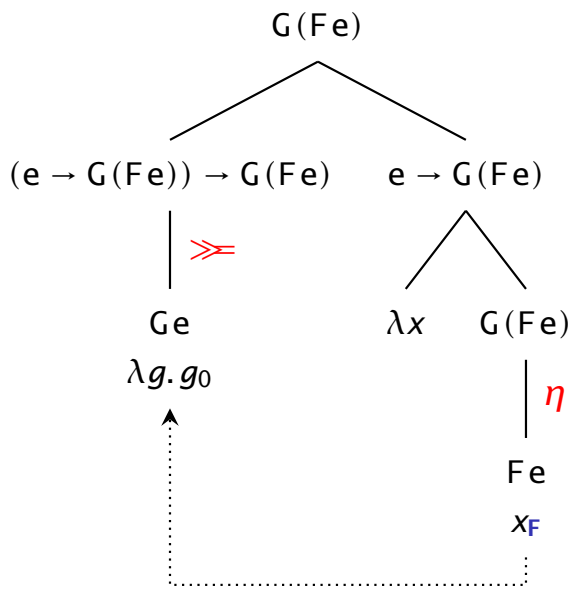
Deriving both layerings

Alternatives over assignments is super easy to derive:

$$\left(\lambda g.\, g_0, \mathbf{alt}_{\lambda g.\, g_0}\right) = (\lambda g.\, g_0)_{\mathbf{F}}$$

Assignments over alternatives is a *little* trickier:

$$\lambda g.\, (g_0, \mathbf{alt}_{g_0}) =$$



Visualizing higher-order meanings

We can represent higher-order meanings (or monadic meanings more generally) using a handy "tower" notation. For example, our two meanings for focused pronouns can be represented as follows:

$$\begin{array}{c} \lambda g.[\,] \\ \hline ([\,], \mathbf{alt}_{g_0}) \\ \hline g_0 \\ \mathsf{G}\,(\mathsf{F}\,\mathsf{e}) \end{array} \qquad \begin{array}{c} ([\,], \mathbf{alt}_{\lambda g.\, g_0}) \\ \hline \lambda g.[\,] \\ \hline g_0 \\ \mathsf{F}\,(\mathsf{G}\,\mathsf{e}) \end{array}$$

When you get down to it, both of these have g_0 as their "core" value. But the layerings of the effects are different, as are the invoked alternatives.

For flexibility?

Having two separate monads, one for focus, and one for pronouns, allows us to generate both layerings of focus and pronominal effects: $\mathsf{G}(\mathsf{F}\,a)$, and $\mathsf{F}(\mathsf{G}\,a)$. Is there evidence that such flexibility is warranted?

Arguments from Jacobson (2000, 2004) suggest it might be:

1. Every third grade boy loves his mother.
   And every FOURTH$_{\mathsf{F}}$ grade boy loves himself.

For flexibility?

Having two separate monads, one for focus, and one for pronouns, allows us to generate both layerings of focus and pronominal effects: $\mathsf{G}(\mathsf{F}\,a)$, and $\mathsf{F}(\mathsf{G}\,a)$. Is there evidence that such flexibility is warranted?

Arguments from Jacobson (2000, 2004) suggest it might be:

1. Every third grade boy loves his mother.
   And every FOURTH$_{\mathsf{F}}$ grade boy loves himSELF$_{\mathsf{F}}$.

For flexibility?

Having two separate monads, one for focus, and one for pronouns, allows us to generate both layerings of focus and pronominal effects: $\mathsf{G}(\mathsf{F}\,a)$, and $\mathsf{F}(\mathsf{G}\,a)$. Is there evidence that such flexibility is warranted?

Arguments from Jacobson (2000, 2004) suggest it might be:

1. Every third grade boy loves his mother.
   And every FOURTH$_\mathsf{F}$ grade boy loves himSELF$_\mathsf{F}$.
2. Every third grade boy loves himself.
   And every FOURTH$_\mathsf{F}$ grade boy loves himself.

For flexibility?

Having two separate monads, one for focus, and one for pronouns, allows us to generate both layerings of focus and pronominal effects: $\mathsf{G}(\mathsf{F}\,a)$, and $\mathsf{F}(\mathsf{G}\,a)$. Is there evidence that such flexibility is warranted?

Arguments from Jacobson (2000, 2004) suggest it might be:

1. Every third grade boy loves his mother.
   And every FOURTH$_\mathsf{F}$ grade boy loves himSELF$_\mathsf{F}$.
2. Every third grade boy loves himself.
   And every FOURTH$_\mathsf{F}$ grade boy loves HIMself$_\mathsf{F}$.

For flexibility?

Having two separate monads, one for focus, and one for pronouns, allows us to generate both layerings of focus and pronominal effects: $\mathsf{G}(\mathsf{F}\,a)$, and $\mathsf{F}(\mathsf{G}\,a)$. Is there evidence that such flexibility is warranted?

Arguments from Jacobson (2000, 2004) suggest it might be:

1. Every third grade boy loves his mother.
   And every FOURTH$_\mathsf{F}$ grade boy loves himSELF$_\mathsf{F}$.
2. Every third grade boy loves himself.
   And every FOURTH$_\mathsf{F}$ grade boy loves HIMself$_\mathsf{F}$.

Jacobson suggests that the *himSELF* accent pattern is associated with (in our terms) $\mathsf{G}(\mathsf{F}\,a)$, and the *HIMself* accent pattern with $\mathsf{F}(\mathsf{G}\,a)$.

Hm...

These arguments seem inconclusive to me. Jacobson does not actually give a formal account of how the $G(F\,a)$ meanings are supposed to work, and this isn't trivial!

In particular, quantifiers don't know how to scope over focused meanings, so this implies a $\sim$ operator in the scope of, e.g., *every*. But it's far from obvious how $\sim$'s should interact with bound things!

More generally, one wonders why the following wouldn't work:

$$\text{himSELF}_F \rightsquigarrow \{\lambda g.\, g_0, \lambda g.\, \mathbf{mom}\, g_0, \ldots\}$$
$$\text{HIMself}_F \rightsquigarrow \{\lambda g.\, g_0, \lambda g.\, g_1, \ldots\}$$

Summing up

With focus and pronouns, there's little reason to assume a combined monad. It is enough to toss both monads into your grammar and let the chips fall where they may.

The two-monads approach predicts two substantively different layerings of focus and pronominal effects. It is unclear whether there is independent evidence for this.

Obviously, this is all very preliminary, pending further investigation. . .

Functors (and structured meanings)

Monads, etc

We have seen several constructs for characterizing how "enriched" meanings interact with others in semantic composition.

- **Monads** allow us to sequence an enriched thing with a scope:

$$\gg\!= :: T\,a \to (a \to T\,b) \to T\,b$$

- **Applicatives** give rise to a notion of enriched functional application:

$$\blacktriangleright :: T\,a \to T\,(a \to b) \to T\,b$$

- **Functors** allow us to map a function over some structure:

$$\triangleright :: T\,a \to (a \to b) \to T\,b$$

Relative strength

Every monad is applicative, and every applicative is a functor:

Monad $\subset$ **Applicative** $\subset$ **Functor**

To wit, given a monadic $\gg\!=$, $\blacktriangleright$ and $\triangleright$ can be recovered as follows:

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

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

Against monadic (and applicative) treatments of focus

A sentence like *JOHN$_F$ saw MARY$_F$* is only felicitous as a response to a question like *who saw whom?* or a sentence like *Bill saw Sue*.

If you try using *JOHN$_F$ saw MARY$_F$* to reply to *who saw Mary?* or *Bill saw Mary*, over-focusing results in infelicity.

It is difficult to see how this can be ruled out with a type $\mathsf{F}\,\mathsf{t}$ meaning:

$$(\mathbf{saw}\,\mathbf{m}\,\mathbf{b}, \{\mathbf{saw}\,y\,x \mid x \in \mathbf{alt_b}, y \in \mathbf{alt_m}\})$$

But this is precisely the kind of meaning generated in monadic (as well as applicative) approaches to F!

Higher-order meanings to the rescue?

Higher-order meanings are finer-grained than "flat" focus structures:

$$\Big((\mathbf{saw}\,\mathbf{m}\,\mathbf{j}, \{\mathbf{saw}\,x\,\mathbf{j} \mid x \in \mathbf{alt_m}\}), \Big\{(\mathbf{saw}\,\mathbf{m}\,y, \{\mathbf{saw}\,x\,y \mid x \in \mathbf{alt_m}\}) \mid y \in \mathbf{alt_j}\Big\}\Big)$$

This extra resolution may allow us to rule out over-focusing (if interested, you can see a sketchy implementation here).

But for this to work, higher-order meanings have to be the *only* option (since flat meanings illicitly allow over-focusing).

Making higher-order meanings a default

If $\gg\!=$ is replaced with F's mapping operation $\triangleright$, higher-order meanings with multiply-focused constructions will be the only option:

$$(x, X) \triangleright f := (f\,x, \{f\,x' \mid x' \in X\})$$

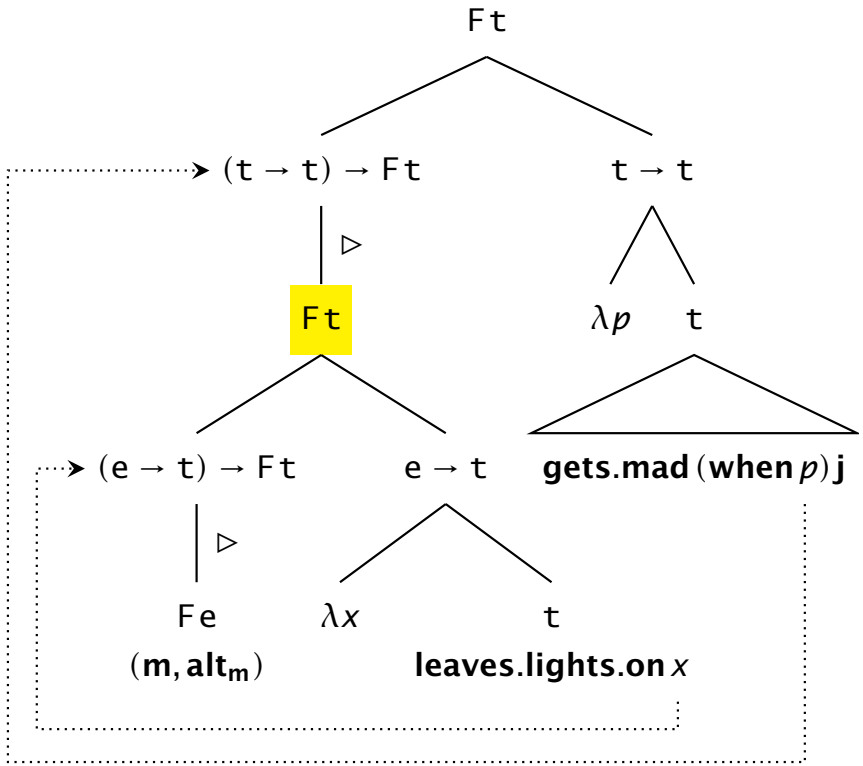
A schematic example:

$$m \triangleright \lambda x.\, n \triangleright \lambda y. \ldots\, x \ldots\, y \ldots :: \mathrm{F}\,(\mathrm{F}\,a)$$

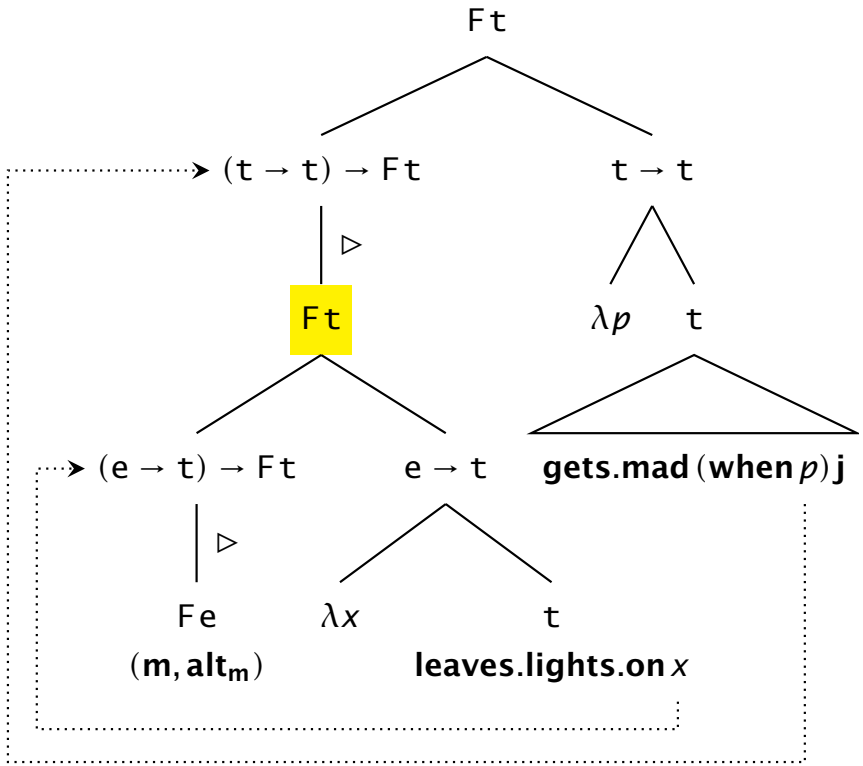
Still predicts island insensitivity (and, naturally, selectivity):

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

Sample island derivation: missing η's!



Sample island derivation: missing η's!

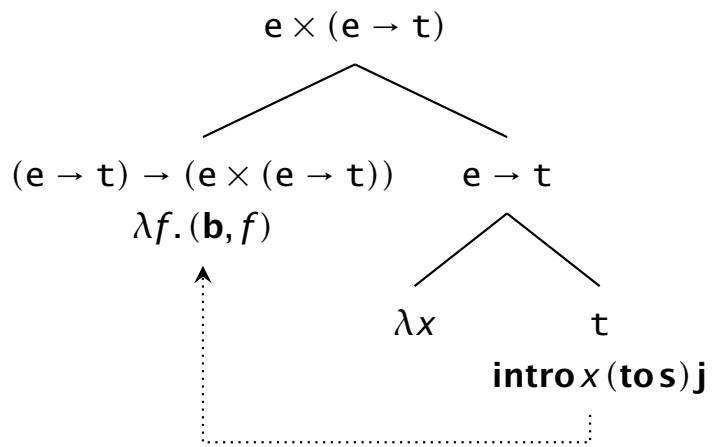


$$(\mathbf{mad}\,(\mathbf{when}\,(\mathbf{llo}\,\mathbf{m}))\,\mathbf{j},\ \{\mathbf{mad}\,(\mathbf{when}\,(\mathbf{llo}\,x))\,\mathbf{j} \mid x \in \mathbf{alt_m}\})$$

Reminder about structured meanings

Here is one simple way to implement structured meanings:

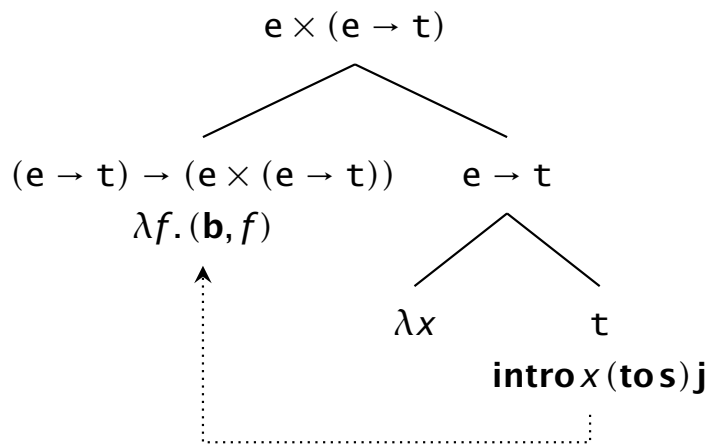
$$\cdot_{\mathbf{F}} :: a \to (a \to b) \to (a \times (a \to b))$$
$$\cdot_{\mathbf{F}} := \lambda x.\lambda f.(x, f)$$



Reminder about structured meanings

Here is one simple way to implement structured meanings:

$$\cdot_{\mathbf{F}} :: a \to (a \to b) \to (a \times (a \to b))$$
$$\cdot_{\mathbf{F}} := \lambda x.\lambda f.(x, f)$$



$$(\mathbf{b}, \lambda x.\mathbf{intro}\, x\, (\mathbf{to}\, \mathbf{s})\, \mathbf{j})$$

An alternative: functors for structured meanings

As long as we're using functors for focus, it's cool to note that structured meaning composition can be characterized in terms of functors.

First, a type constructor ('K' for Krifka):

$$\mathrm{K}_b\, a ::= (b, b \to a)$$

Then, a mapping operation:

$$\triangleright \quad :: \mathrm{K}_b\, a \to (a \to c) \to \mathrm{K}_b\, c$$
$$(x, f) \triangleright g := $$

An alternative: functors for structured meanings

As long as we're using functors for focus, it's cool to note that structured meaning composition can be characterized in terms of functors.

First, a type constructor ('K' for Krifka):

$$\mathrm{K}_b\, a ::= (b, b \to a)$$

Then, a mapping operation:

$$\begin{aligned} \triangleright \quad &:: \mathrm{K}_b\, a \to (a \to c) \to \mathrm{K}_b\, c \\ (x, f) \triangleright g &:= (x, g \circ f) \end{aligned}$$

F-marks in alternative structured meanings

Then the meaning of F-marks can be given as follows:

$$\cdot_{\mathbf{F}} :: a \to \mathsf{K}_a\, a$$
$$\cdot_{\mathbf{F}} :=$$

F-marks in alternative structured meanings

Then the meaning of F-marks can be given as follows:

$$\cdot_{\mathsf{F}} :: a \to \mathsf{K}_a\, a$$
$$\cdot_{\mathsf{F}} := \lambda x.(x, \lambda y.\, y)$$

We note the following interesting fact:

$$x_{\mathsf{F}}{}^{\triangleright}$$

F-marks in alternative structured meanings

Then the meaning of F-marks can be given as follows:

$$\cdot_{\mathsf{F}} :: a \to \mathsf{K}_a\, a$$
$$\cdot_{\mathsf{F}} := \lambda x.\,(x, \lambda y.\, y)$$

We note the following interesting fact:

$$x_{\mathsf{F}}{}^{\triangleright} = (x, \lambda y.\, y)^{\triangleright}$$

F-marks in alternative structured meanings

Then the meaning of F-marks can be given as follows:

$$\cdot_{\mathsf{F}} :: a \to \mathsf{K}_a\, a$$
$$\cdot_{\mathsf{F}} := \lambda x.(x, \lambda y.\, y)$$

We note the following interesting fact:

$$\begin{aligned} x_{\mathsf{F}}{}^{\triangleright} &= (x, \lambda y.\, y)^{\triangleright} \\ &= \lambda f.(x, f \circ \lambda y.\, y) \end{aligned}$$

F-marks in alternative structured meanings

Then the meaning of F-marks can be given as follows:

$$\cdot_{\mathsf{F}} :: a \to \mathsf{K}_a\, a$$
$$\cdot_{\mathsf{F}} := \lambda x.\,(x, \lambda y.\, y)$$

We note the following interesting fact:

$$\begin{aligned} x_{\mathsf{F}}{}^{\triangleright} &= (x, \lambda y.\, y)^{\triangleright} \\ &= \lambda f.\,(x, f \circ \lambda y.\, y) \\ &= \lambda f.\,(x, f) \end{aligned}$$

Which is just the old structured meanings meaning for F-marked things!

As with monads, the move involves making the semantics more *modular*.

Sample derivations

A sample derivation for *JOHN*$_F$ *left*:

$$\mathbf{j_F} \triangleright \lambda x.\mathbf{left}\, x =$$

Sample derivations

A sample derivation for *JOHN*$_F$ *left*:

$$\mathbf{j_F} \rhd \lambda x.\,\mathbf{left}\,x = (\mathbf{j}, \lambda y.\,y) \rhd \lambda x.\,\mathbf{left}\,x$$

Sample derivations

A sample derivation for *JOHN*$_F$ *left*:

$$\begin{aligned}\mathbf{j_F} \triangleright \lambda x.\,\mathbf{left}\,x &= (\mathbf{j}, \lambda y.\,y) \triangleright \lambda x.\,\mathbf{left}\,x \\ &= (\mathbf{j}, (\lambda x.\,\mathbf{left}\,x) \circ (\lambda y.\,y))\end{aligned}$$

Sample derivations

A sample derivation for *JOHN*$_F$ *left*:

$$\begin{aligned}\mathbf{j_F} \triangleright \lambda x.\,\mathbf{left}\,x &= (\mathbf{j}, \lambda y.\,y) \triangleright \lambda x.\,\mathbf{left}\,x \\ &= (\mathbf{j}, (\lambda x.\,\mathbf{left}\,x) \circ (\lambda y.\,y)) \\ &= (\mathbf{j}, \lambda y.\,\mathbf{left}\,y)\end{aligned}$$

A sample derivation for *JOHN*$_F$ *saw MARY*$_F$:

$$\mathbf{j_F} \triangleright \lambda x.\,\mathbf{m_F} \triangleright \lambda y.\,\mathbf{saw}\,y\,x =$$

Sample derivations

A sample derivation for *JOHN*$_F$ *left*:

$$\begin{aligned}\mathbf{j_F} \triangleright \lambda x.\,\mathbf{left}\,x &= (\mathbf{j}, \lambda y.\,y) \triangleright \lambda x.\,\mathbf{left}\,x \\ &= (\mathbf{j}, (\lambda x.\,\mathbf{left}\,x) \circ (\lambda y.\,y)) \\ &= (\mathbf{j}, \lambda y.\,\mathbf{left}\,y)\end{aligned}$$

A sample derivation for *JOHN*$_F$ *saw MARY*$_F$:

$$\mathbf{j_F} \triangleright \lambda x.\,\mathbf{m_F} \triangleright \lambda y.\,\mathbf{saw}\,y\,x = (\mathbf{j}, \lambda z.\,z) \triangleright \lambda x.\,(\mathbf{m}, \lambda y.\,\mathbf{saw}\,y\,x)$$

Sample derivations

A sample derivation for *JOHN*$_F$ *left*:

$$\begin{aligned}\mathbf{j_F} \triangleright \lambda x.\,\mathbf{left}\,x &= (\mathbf{j}, \lambda y.\,y) \triangleright \lambda x.\,\mathbf{left}\,x \\ &= (\mathbf{j}, (\lambda x.\,\mathbf{left}\,x) \circ (\lambda y.\,y)) \\ &= (\mathbf{j}, \lambda y.\,\mathbf{left}\,y)\end{aligned}$$

A sample derivation for *JOHN*$_F$ *saw MARY*$_F$:

$$\begin{aligned}\mathbf{j_F} \triangleright \lambda x.\,\mathbf{m_F} \triangleright \lambda y.\,\mathbf{saw}\,y\,x &= (\mathbf{j}, \lambda z.\,z) \triangleright \lambda x.\,(\mathbf{m}, \lambda y.\,\mathbf{saw}\,y\,x) \\ &= (\mathbf{j}, (\lambda x.\,(\mathbf{m}, (\lambda y.\,\mathbf{saw}\,y\,x))) \circ (\lambda z.\,z))\end{aligned}$$

Sample derivations

A sample derivation for *JOHN*$_F$ *left*:

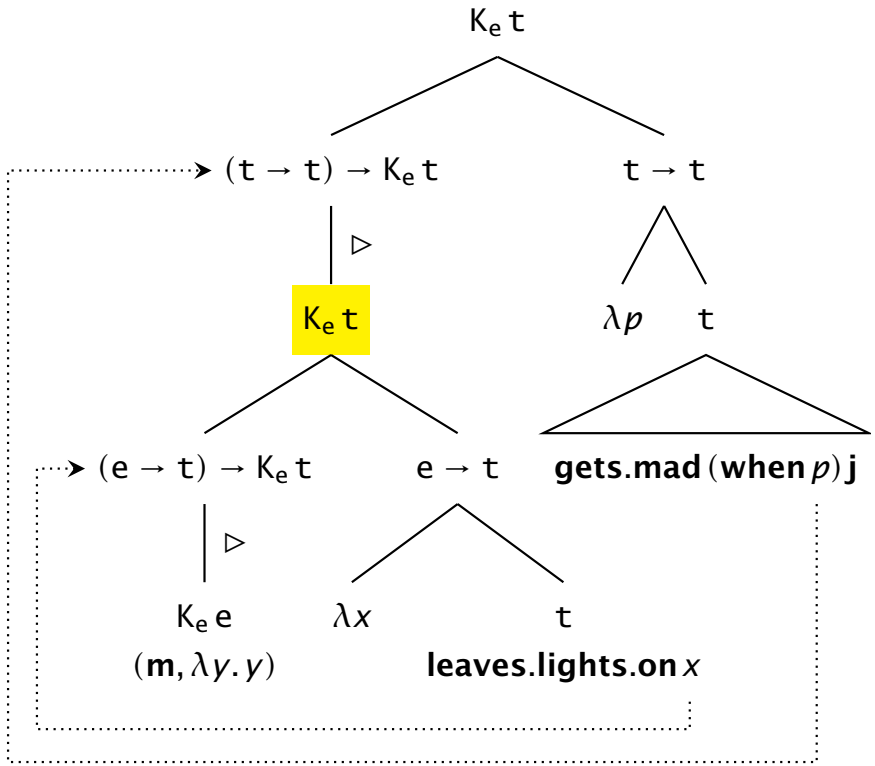
$$\begin{aligned}\mathbf{j_F} \rhd \lambda x.\,\mathbf{left}\,x &= (\mathbf{j}, \lambda y.\,y) \rhd \lambda x.\,\mathbf{left}\,x \\ &= (\mathbf{j}, (\lambda x.\,\mathbf{left}\,x) \circ (\lambda y.\,y)) \\ &= (\mathbf{j}, \lambda y.\,\mathbf{left}\,y)\end{aligned}$$

A sample derivation for *JOHN*$_F$ *saw MARY*$_F$:

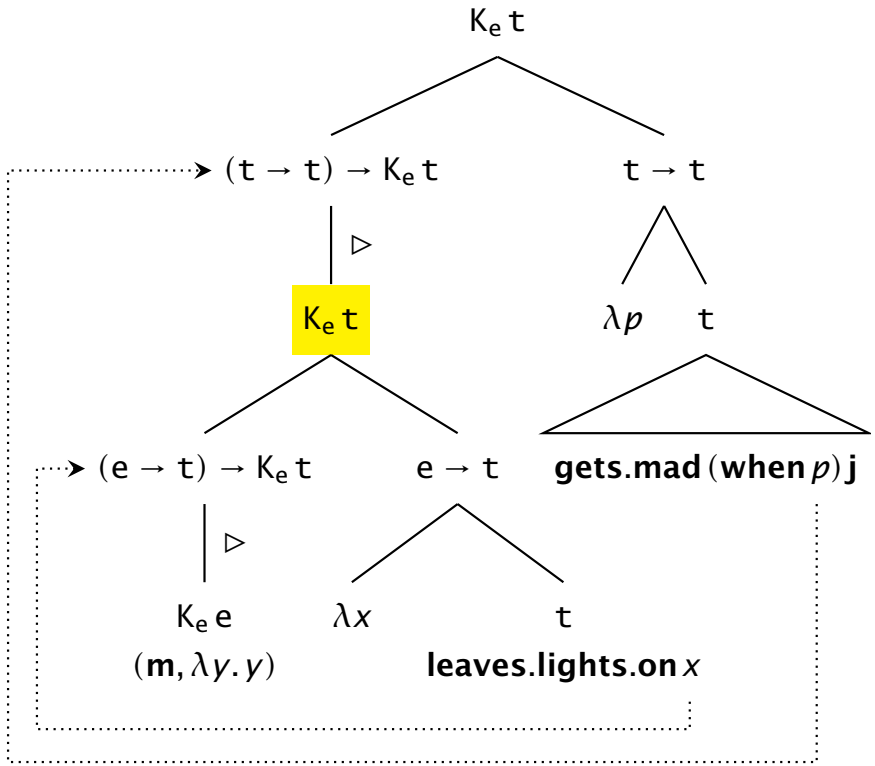
$$\begin{aligned}\mathbf{j_F} \rhd \lambda x.\,\mathbf{m_F} \rhd \lambda y.\,\mathbf{saw}\,y\,x &= (\mathbf{j}, \lambda z.\,z) \rhd \lambda x.\,(\mathbf{m}, \lambda y.\,\mathbf{saw}\,y\,x) \\ &= (\mathbf{j}, (\lambda x.\,(\mathbf{m}, (\lambda y.\,\mathbf{saw}\,y\,x))) \circ (\lambda z.\,z)) \\ &= (\mathbf{j}, \lambda z.\,(\mathbf{m}, \lambda y.\,\mathbf{saw}\,y\,z))\end{aligned}$$

[Can be flattened into $((\mathbf{j}, \mathbf{m}), \lambda(z, y).\,\mathbf{saw}\,y\,z)$]

Evading islands



Evading islands



$$(\mathbf{m},\, \lambda y.\, \mathbf{mad}\,(\mathbf{when}\,(\mathbf{llo}\,y))\,\mathbf{j})$$

Interaction and selectivity

Because the focus apparatus is separate from the apparatus for binding, we'll still predict two possible ways for focus and binding to interact. This holds for functorial Rooth, as well as functorial Krifka.

Higher-order meanings are the *default* with functors (when you have more than one effect-ful thing). So selectivity is predicted, as well.

Over-focusing

As Krifka emphasizes, representations like the following can be used to directly state question-answer congruence, in a way that rules out over-focusing in the answer:

$$((\mathbf{j}, \mathbf{m}), \lambda(z, y).\,\mathbf{saw}\,y\,z)$$

This fits most naturally with a structured-meaning account of questions, and is a little clunky to adapt to over-focusing in non-QA contexts.

Which construct, and when?

Relative strength, redux

Every monad is applicative, and every applicative is a functor:

Monad $\subset$ **Applicative** $\subset$ **Functor**

To wit, given a monadic $\gg=$, $\blacktriangleright$ and $\triangleright$ can be recovered as follows:

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

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

So how do you know which one you need?

Monads are a natural fit for indefinites and dynamics. Functors are a natural fit for focus. Why?

It turns on a funny thing we'll call an "impure arrow". For any functorial, applicative, or monadic T, an impure arrow is something with type:

$$a \to T\, b$$

Impure arrows and scope

Impure arrows are important because *scope-taking creates them*. Monads tell us how this process can result in a "flat" computation:

$$(\gg\!=) :: T\,a \to (a \to T\,b) \to T\,b$$

By contrast, functors and applicatives inevitably force us to build up higher-order $T\,(T\,b)$ things from impure arrows!

Impure arrows and choice

So the choice of functor, applicative, or monad turns on a few things. For example, does a $T\,a$ ever need to scope over a $T\,b$?

- If so, do you need the ability to return a $T\,b$? Then use a **monad**!
- If no, do you need the ability to *ever* return a "flat" computation when you have two fancy things? Then use an **applicative**!
- If you're happy with always returning higher-order things, or you have a way to deal with them (e.g., $\sim$), a **functor** may fit the bill!

Other considerations will of course be relevant. For example, our structured meanings functor is neither monadic nor applicative!

Some fun stuff

Prolegomenon to free monads

Here's an interesting recursive type:

$$\mathtt{Functor}\, f \Longrightarrow \mathbb{F}_f\, a ::= \underbrace{a}_{\mathtt{Pure}} \mid \overbrace{f\,(\mathbb{F}_f\, a)}^{\mathtt{Impure}}$$

A concrete example, with $f ::= \mathsf{S}$:

$$\mathbb{F}_{\mathsf{S}}\, \mathsf{e} ::= \mathbf{a} \mid \mathbf{b} \mid \{\mathbf{a}\} \mid \{\mathbf{a}, \mathbf{b}\} \mid \{\{\mathbf{a}\}, \{\mathbf{b}\}\} \mid \ldots$$

[Really, the two branches in our recursive data type should be tagged with `Pure` and `Impure` *data constructors*, but I'm glossing over this complication here.]

Free monads

When f is a functor, $\mathbb{F}_f$ forms a monad, the **Free monad** over f!

$$\eta\, x := x$$

$$m \gg\!= f := \begin{cases} f\, m & \text{if } m \text{ is Pure} \\ m \triangleright (\lambda x.\, x \gg\!= f) & \text{if } m \text{ is Impure} \end{cases}$$

$$\begin{aligned} \{0,1\} \gg\!= \lambda x.\, \{3,4\} \gg\!= \lambda y.\, \eta\,(x+y) &= \{0,1\} \gg\!= \lambda x.\, \{3,4\} \gg\!= \lambda y.\, x+y \\ &= \{0,1\} \gg\!= \lambda x.\, \{x+3,\, x+4\} \\ &= \{\{3,4\},\, \{4,5\}\} \end{aligned}$$

So free monads are really functor-like in their bones: they obligatorily yield higher-order meanings in the presence of multiple “impure” things.

Comonads

Monads arise from two functions:

$$\eta :: a \to T\,a \qquad\qquad \gg\!= :: T\,a \to (a \to T\,b) \to T\,b$$

If we "reverse the arrows", we're in the presence of a **comonad**:

$$\epsilon :: a \leftarrow T\,a \qquad\qquad \chi :: T\,a \leftarrow (a \leftarrow T\,b) \leftarrow T\,b$$

. . . Or, more familiarly:

$$\epsilon :: T\,a \to a \qquad\qquad \chi :: T\,b \to (T\,b \to a) \to T\,a$$

Comonadic focus

Though structured meanings aren't monadic, they *are* comonadic!

$$\begin{aligned} \epsilon &:: \mathsf{K}_b\, a \to a \\ \epsilon\,(x, f) &:= f\, x \end{aligned}$$

$$\begin{aligned} \chi &:: \mathsf{K}_b\, a \to (\mathsf{K}_b\, a \to c) \to \mathsf{K}_b\, c \\ \chi\,(x, f) &:= \lambda g.\,(x, \lambda y.\, g\,(y, f)) \end{aligned}$$

A couple neat things about this formulation (aka the **Store** comonad):

- Comonads are functors, which means we still have access to $\triangleright$!
- ϵ looks a lot like the focus interpretation operator $\sim$!
- χ allows us to preserve a focus after association with $\sim$ or **only**!

References I

Jacobson, Pauline. 2000. Paychecks, stress, and variable-free semantics. In Brendan Jackson & Tanya Matthews (eds.), *Proceedings of Semantics and Linguistic Theory 10*, 65–82. Ithaca, NY: Cornell University. http://dx.doi.org/10.3765/salt.v10i0.3103.

Jacobson, Pauline. 2004. Kennedy's puzzle: What I'm named or who I am?. In Kazuha Watanabe & Robert B. Young (eds.), *Proceedings of Semantics and Linguistic Theory 14*, 145–162. Ithaca, NY: Cornell University. http://dx.doi.org/10.3765/salt.v14i0.2911.

Rooth, Mats. 1985. *Association with focus*. University of Massachusetts, Amherst Ph.D. thesis.