Notes on effect driven interpretation

Alex Shilen

November 4, 2025

1 What's an effect?

- Consider a simple composition system, roughly that of Heim and Kratzer (1998), with lexical entries and composition operations that together form a type driven grammar.

Types:

$$
\begin{array}{llr}
\tau ::= \mathsf{e} \mid \mathsf{t} \mid \cdots & & \text{Base types} \\
\quad\mid & \tau \to \tau & \text{Function types} \\
\quad\mid & \tau \times \tau & \text{Product types} \\
\quad\mid & \tau + \tau & \text{Sum types} \\
\quad\mid & \{\tau\} & \text{Set types}
\end{array}
$$

Modes of combination:

$$
\begin{array}{lr}
(\mathbin{>}) :: (\alpha \to \beta) \to \alpha \to \beta & \text{Forward Application} \\
f > x \coloneqq f\,x &
\end{array}
$$

$$
\begin{array}{lr}
(\mathbin{<}) :: \alpha \to (\alpha \to \beta) \to \beta & \text{Backward Application} \\
x < f \coloneqq f\,x &
\end{array}
$$

$$
\begin{array}{lr}
(\circ) :: (\beta \to \varsigma) \to (\alpha \to \beta) \to \alpha \to \varsigma & \text{Function Composition} \\
f \circ g \coloneqq \lambda x.\, f\,(g\,x) &
\end{array}
$$

$$
\begin{array}{lr}
(\sqcap) :: (\mathsf{e} \to \mathsf{t}) \to (\mathsf{e} \to \mathsf{t}) \to \mathsf{e} \to \mathsf{t} & \text{Predicate Modification} \\
f \sqcap g \coloneqq \lambda x.\, f\,x \wedge g\,x &
\end{array}
$$

$$
\begin{array}{lr}
(\restriction) :: (\alpha \to \beta \to \mathsf{t}) \to (\beta \to \mathsf{t}) \to \alpha \to \beta \to \mathsf{t} & \text{Relation Restriction} \\
r \restriction p \coloneqq \lambda x \lambda y.\, p\,y \wedge r\,x\,y &
\end{array}
$$

- Without saying more, certain phenomena are difficult to model with this basic framework:

(1)
- context dependence
- presupposition
- supplemental content
- nondeterminism
- quantification
- focus with association
- topicality
- dynamism

- These phenomena have something in common. Each contributes additional meaning to consituents that would otherwise be type e. We might express this additional meaning as follows.

Expression	Type	Denotation
it	$\mathtt{i} \to \mathtt{e}$	$\lambda i.\, \pi\, i$
the planet	$\mathtt{e} + \bot$	x if $\textbf{planet} = \{x\}$ else #
Jupiter, a planet	$\mathtt{e} \times \mathtt{t}$	$\langle \mathbf{j}, \textbf{planet}\, \mathbf{j} \rangle$
which planet	$\{\mathtt{e}\}$	$\{x \mid \textbf{planet}\, x\}$
no planet	$(\mathtt{e} \to \mathtt{t}) \to \mathtt{t}$	$\lambda Q.\, \neg \exists x.\, \textbf{planet}\, x \wedge Q\, x$
JUPITER	$\mathtt{e} \times \{\mathtt{e}\}$	$\langle \mathbf{j}, \{x \mid x \sim \mathbf{j}\} \rangle$
as for Jupiter	$\mathtt{s} \to (\mathtt{e} \times \mathtt{s})$	$\lambda s.\, \langle \mathbf{j}, \mathbf{j} \mathbin{+\!\!+} s \rangle$
a planet	$\mathtt{s} \to \{\mathtt{e} \times \mathtt{s}\}$	$\lambda s.\, \{\langle x, x \mathbin{+\!\!+} s \rangle \mid \textbf{planet}\, x\}$

- The basic intuition to develop here is that in each example objects of type e are **embedded in some additional structure**. We will call this additional structure an **effect**.

2 Functors

- We can say more. Because it's possible to define functions that transform the embedded objects of type e while preserving the surrounding structure, each of these phenomena is an example of a **functor**.

- We will call such structure-preserving transformations **maps** and refer to them in symbols as $\bullet$. Here are the maps for the environment sensitive and nondeterminate structures, for any type α.

(2) $f \bullet x = \lambda i. f (x\ i)$ $\qquad (e \to \alpha) \to (i \to e) \to (i \to \alpha)$

(3) $f \bullet x = \{f\ y \mid y \in x\}$ $\qquad (e \to \alpha) \to \{e\} \to \{\alpha\}$

- We formalize the intuition of structure preservation by requiring maps to satisfy two laws.

(4) Identity

$$id \bullet x = x$$

(5) Composition

$$f \bullet (g \bullet x) = (f \circ g) \bullet x$$

- In English: mapping the identity function does nothing, while mapping a composite function $f \circ g$ is the same as first mapping g and then mapping f.

2.1 Putting maps to work

- Why is it worthwhile to observe that these phenomena give rise to functors? B&C observe that their maps can be used to construct composition operations that allow effectful constituents to compose where simple function application will not. First some more notational conventions: we will define type constructors for the different effects. The type constructors will serve as names for their underlying functors.

$$\begin{array}{ll}
\boxed{\mathtt{R}\,\alpha} ::= \mathtt{i} \to \alpha & \boxed{\mathtt{C}\,\alpha} ::= (\alpha \to \mathtt{t}) \to \mathtt{t} \\
\boxed{\mathtt{W}\,\alpha} ::= \alpha \times \mathtt{t} & \boxed{\mathtt{F}\,\alpha} ::= \alpha \times \{\alpha\} \\
\boxed{\mathtt{M}\,\alpha} ::= \alpha + \bot & \boxed{\mathtt{S}\,\alpha} ::= \{\alpha\} \\
\boxed{\mathtt{T}\,\alpha} ::= \mathtt{s} \to (\alpha \times \mathtt{s}) & \boxed{\mathtt{D}\,\alpha} ::= \mathtt{s} \to \{\alpha \times \mathtt{s}\}
\end{array}$$

- We will identify each phenomenon in (1) with a functor.

Expression	Type	Denotation
it	$\boxed{\mathtt{R\,e}}$	$\lambda i.\, \pi\, i$
the planet	$\boxed{\mathtt{M\,e}}$	x if $\textbf{planet} = \{x\}$ else #
Jupiter, a planet	$\boxed{\mathtt{W\,e}}$	$\langle \mathbf{j}, \textbf{planet}\,\mathbf{j}\rangle$
which planet	$\boxed{\mathtt{S\,e}}$	$\{x \mid \textbf{planet}\,x\}$
no planet	$\boxed{\mathtt{C\,e}}$	$\lambda Q.\, \neg\exists x.\, \textbf{planet}\,x \wedge Q\,x$
JUPITER	$\boxed{\mathtt{F\,e}}$	$\langle \mathbf{j}, \{x \mid x \sim \mathbf{j}\}\rangle$
as for Jupiter	$\boxed{\mathtt{T\,e}}$	$\lambda s.\, \langle \mathbf{j}, \mathbf{j} +\!\!\!+\, s\rangle$
a planet	$\boxed{\mathtt{D\,e}}$	$\lambda s.\, \{\langle x, x +\!\!\!+\, s\rangle \mid \textbf{planet}\,x\}$

- Now we can add forward and backward maps to the grammar, for arbitrary functors Σ.

Combinators

$$(>) \;::\; (\alpha \to \beta) \to \alpha \to \beta \qquad \text{Forward Application}$$
$$f > a \coloneqq f\,a$$

$$(<) \;::\; \alpha \to (\alpha \to \beta) \to \beta \qquad \text{Backward Application}$$
$$a < f \coloneqq f\,a$$

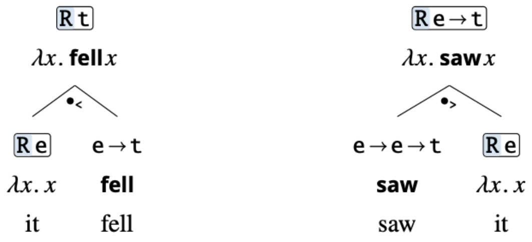
$$\vdots \qquad \text{Other Basic Combinators}$$

$$(\bullet_{>}) \;::\; (\alpha \to \beta) \to \boxed{\Sigma\,\alpha} \to \boxed{\Sigma\,\beta} \qquad \text{Forward Map}$$
$$f \bullet_{>} A \coloneqq f \bullet A$$

$$(\bullet_{<}) \;::\; \boxed{\Sigma\,\alpha} \to (\alpha \to \beta) \to \boxed{\Sigma\,\beta} \qquad \text{Backward Map}$$
$$A \bullet_{<} f \coloneqq f \bullet A$$

- Here's the immediate payoff: the maps can be used to compose simple functions with effectful arguments. We just map the functions over the embedded arguments.

(6)



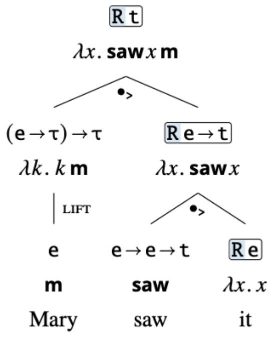
- We face an immediate problem: there's no way to compose functions that are themselves effectful, as e.g. in (7), where *saw it* is a context dependent predicate.

(7) Mary saw it.

- One approach is to introduce Partee's LIFT operation, and instead of mapping a function over an argument, map the argument over the function.

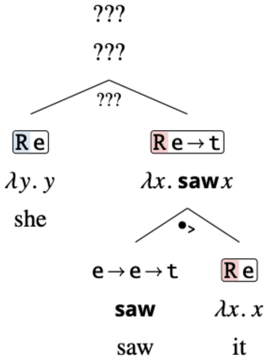
(8) $\textsc{lift} = \lambda x \lambda k.k\;x$ $\qquad \alpha \to (\alpha \to \tau) \to \tau$

(9)



- Next problem: it appears that we still can't compose two siblings when both have effects but are otherwise composable.

(10)



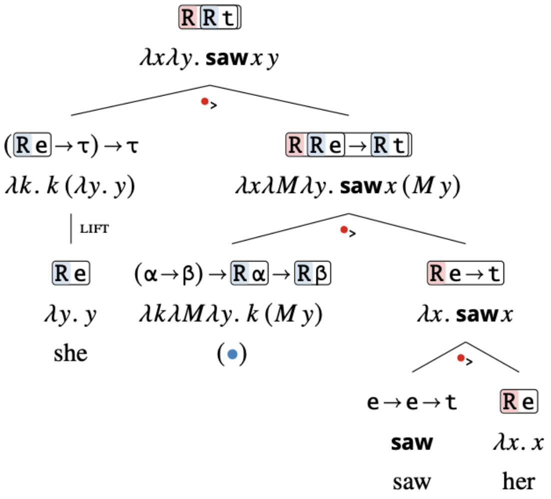
2.2 Higher Order Effects

- B&C remark: "The first step in redressing this unfortunate incomposability is deciding what sort of thing (10) ought to denote." One option is to interpret each pronoun as a separate request for an antecedent. Then the meaning of *she saw it* might be of type $i \rightarrow i \rightarrow t$, or RRt. B&C call such a nesting of effects **higher order**.

(11) $[\![\text{she saw it}]\!] = \lambda x \lambda y.\textbf{saw}\ x\ y$ $\qquad RRt$

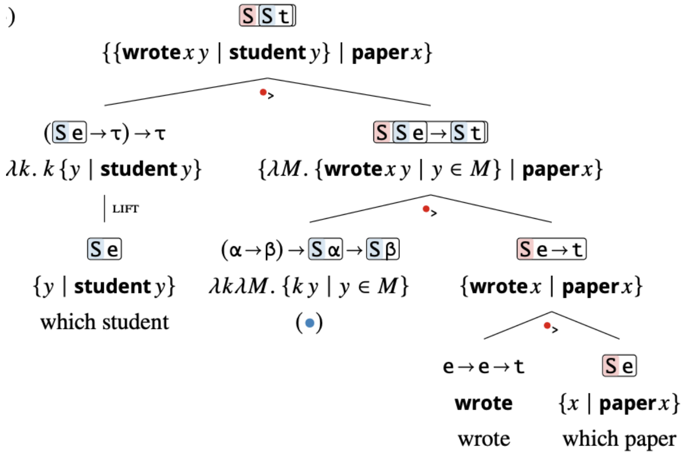
- With some cleverness, it's possible to derive this interpretation with the ingredients we have. We "map the map".

(12)



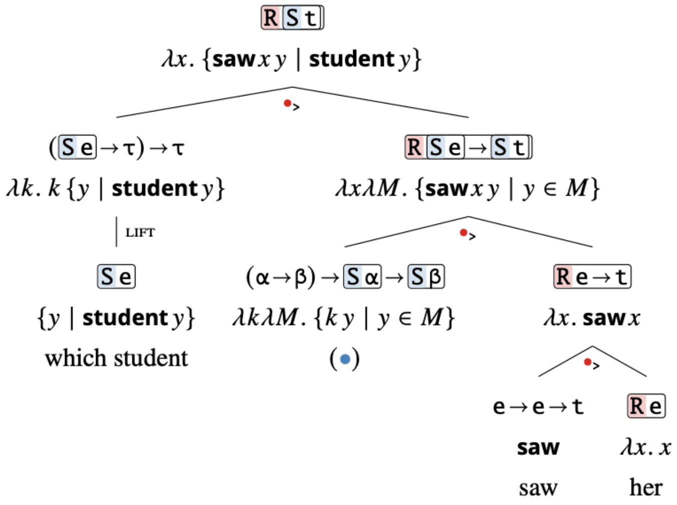
- “Because all of the effects we’ve introduced are functorial, and the only interesting aspects of the derivation are the maps, the tree is a template for composition with any of the enriched meanings. For instance, switching R for S immediately derives a multiple-wh question”

(13)



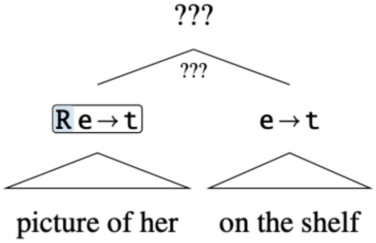
- This even works when function and argument have different effects.

(14)



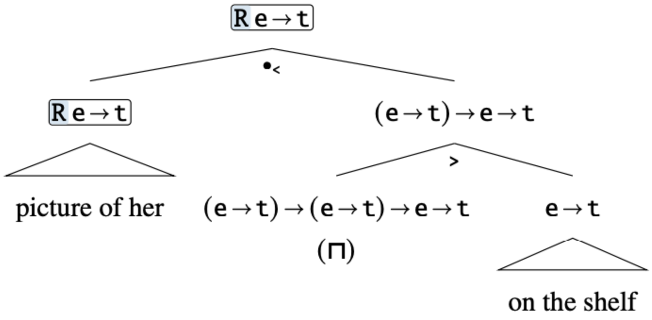
- Now, what about modes of composition beyond function application, e.g. predicate modification?

(15)

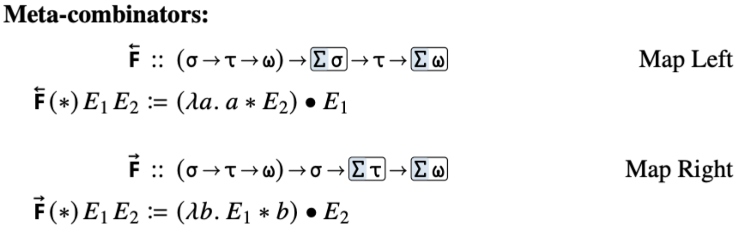


- We might pursue a similar strategy and introduce a PM morpheme.

(16)



- But there are reasons not to go down this path: "In this manner, eventually all modes of combination will need to be realized lexically. Whether this is syntactically justifiable is open to debate, but it certainly increases the distance between the forms that are uttered and the forms that are interpreted. And as a practical matter, the resulting combinatorial system is admittedly unwieldy."
- Instead, B&C propose a more systematic approach. We introduce a transformation on composition operations, so that "in general, if there is a mode $*$ that can combine constituents $M :: \sigma$ and $N :: \tau$, then there is also a mode to combine constituents $M :: \sigma$ and $N :: \Sigma\tau$, provided Σ is a functor."



Meta-combinators:

$$\overleftarrow{\mathsf{F}} :: (\sigma \to \tau \to \omega) \to \boxed{\Sigma\,\sigma} \to \tau \to \boxed{\Sigma\,\omega} \qquad \text{Map Left}$$

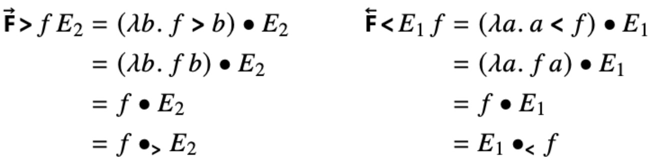
$$\overleftarrow{\mathsf{F}}(*)\,E_1\,E_2 \coloneqq (\lambda a.\, a * E_2) \bullet E_1$$

$$\overrightarrow{\mathsf{F}} :: (\sigma \to \tau \to \omega) \to \sigma \to \boxed{\Sigma\,\tau} \to \boxed{\Sigma\,\omega} \qquad \text{Map Right}$$

$$\overrightarrow{\mathsf{F}}(*)\,E_1\,E_2 \coloneqq (\lambda b.\, E_1 * b) \bullet E_2$$

- We no longer need the bidirectional maps in the grammar, as they can be derived from function application with these meta combinators.

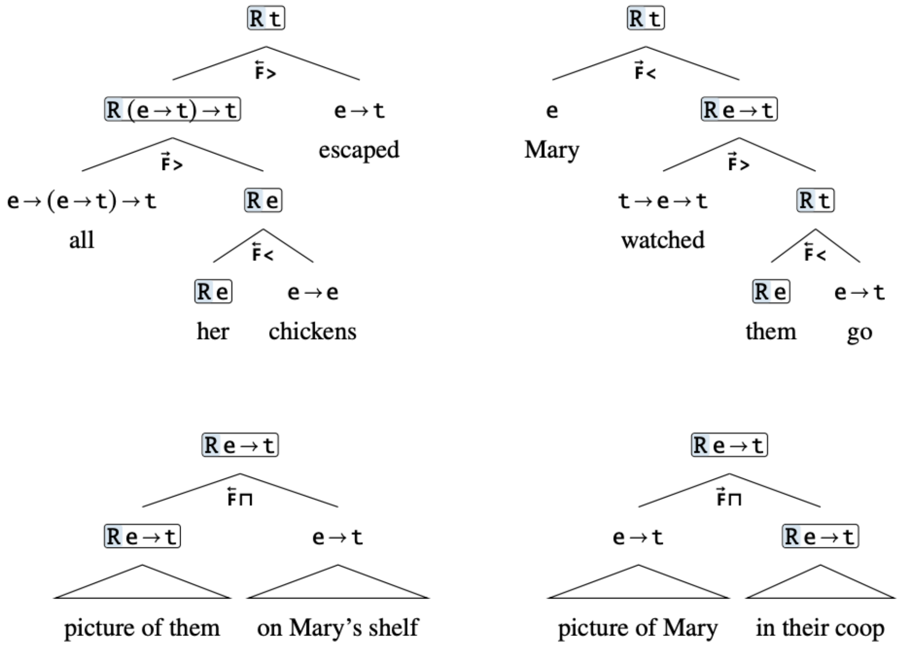
(17)



$$\begin{aligned} \overrightarrow{\mathsf{F}}\,\mathbf{>}\,f\,E_2 &= (\lambda b.\, f \mathbin{\mathbf{>}} b) \bullet E_2 \\ &= (\lambda b.\, f\,b) \bullet E_2 \\ &= f \bullet E_2 \\ &= f \bullet_{>} E_2 \end{aligned} \qquad \begin{aligned} \overleftarrow{\mathsf{F}}\,\mathbf{<}\,E_1\,f &= (\lambda a.\, a \mathbin{\mathbf{<}} f) \bullet E_1 \\ &= (\lambda a.\, f\,a) \bullet E_1 \\ &= f \bullet E_1 \\ &= E_1 \bullet_{<} f \end{aligned}$$

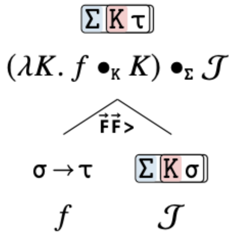
- With the basic composition operations from H&K and the meta combinators, we can handle FA, PM, & etc.

(18)



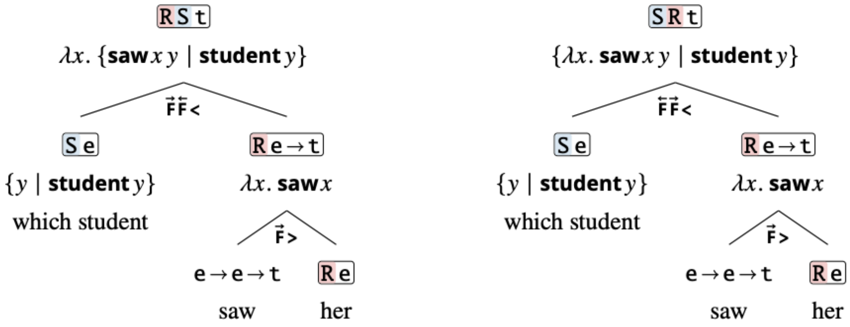
- Interestingly, the $\overrightarrow{F}$ and $\overleftarrow{F}$ combinators can apply to their own output. When the arrows go in the same direction, we're able to map over the underlying object(s) of a doubly effectful constituent. In other words, we derive a composite map.

(19)


Σ K τ
$(\lambda K.\, f \bullet_{K} K) \bullet_{\Sigma} \mathcal{J}$
$\overrightarrow{F}\overrightarrow{F}>$
σ→τ
Σ K σ
f
$\mathcal{J}$

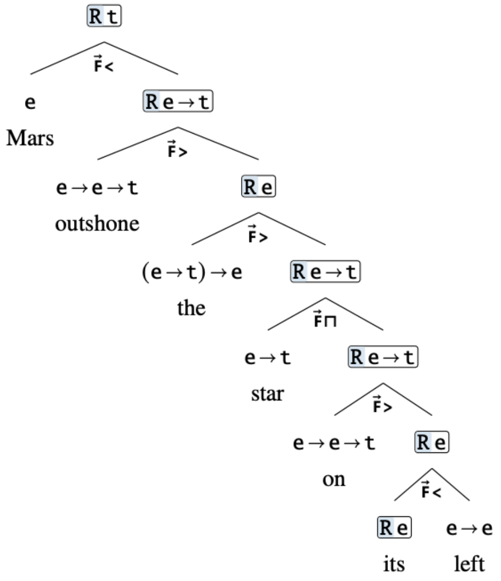
- When the arrows go in opposite directions, we flip the order of effects.

(20)


R S t
$\lambda x.\, \{\mathbf{saw}\, x\, y \mid \mathbf{student}\, y\}$
$\overrightarrow{F}\overleftarrow{F}<$
S e
$\{y \mid \mathbf{student}\, y\}$
which student
R e→t
$\lambda x.\, \mathbf{saw}\, x$
$\overrightarrow{F}>$
e→e→t
saw
R e
her
S R t
$\{\lambda x.\, \mathbf{saw}\, x\, y \mid \mathbf{student}\, y\}$
$\overleftarrow{F}\overrightarrow{F}<$
S e
$\{y \mid \mathbf{student}\, y\}$
which student
R e→t
$\lambda x.\, \mathbf{saw}\, x$
$\overrightarrow{F}>$
e→e→t
saw
R e
her

- A final note on the grammar so far: effects always take widest scope, since our composition operations always map “underneath” them. In (21), for instance, the deeply embedded pronoun forces the entire sentence to be context dependent.

(21)



- The same will be true of each effect: no matter how it’s embedded, it will force the entire sentence to take its type. In particular, effects will not be sensitive to **island boundaries**. And this is mostly empirically justified, as all of the phenomena we’re considering – other than quantifiers – exhibit exceptional scope.

(22) Who remembers when who left?

(23) Mary only gets made when JOHN leaves the lights on.

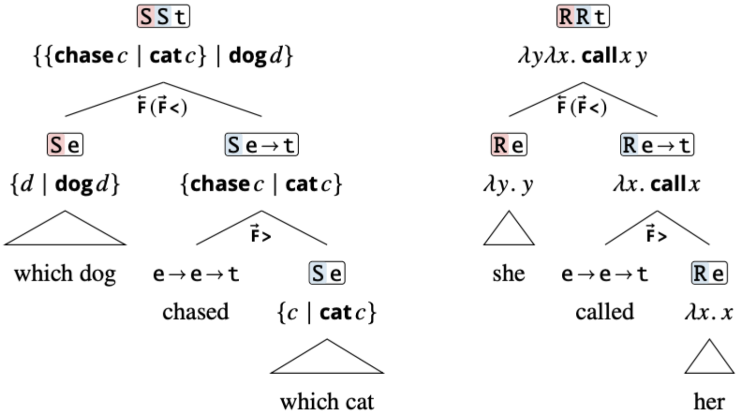
(24) Mary hopes that because her cat, named Sassy, is home, John is too.

(25) Mary’s being out of town means that if you don’t see John’s car, you can be sure nobody’s home.

3 Applicative Functors

- Let's reconsider the higher order denotations we settled for above. Ideally, we'd like to derive meanings closer to what e.g. Hamblin (1976) and H&K derive, where the presence of multiple effects, as long as they're the same type, do not force the entire sentence in which they occur to be higher order.

(26)



3.1 Merging effects

- It's straightforward to equip our grammar with generalized-to-the-worst-case pointwise and assignment sensitive composition operations that **merge** the effects of siblings at each composite node.

(27) Pointwise and Assignment Sensitive Combinators

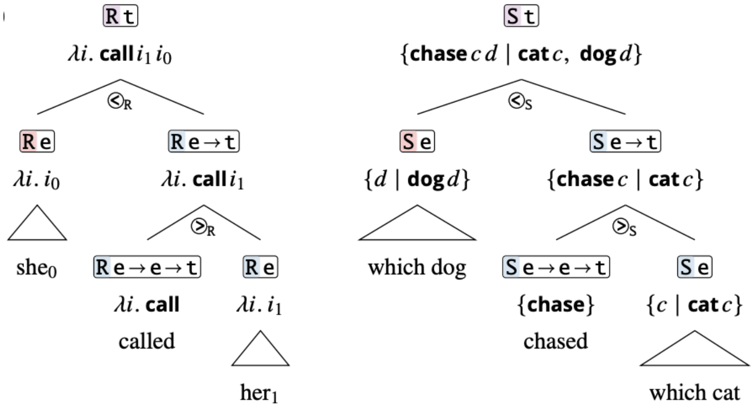
$$(\ogreaterthan_{\mathsf{R}}) :: \boxed{\mathsf{R}\,\alpha\to\beta}\to\boxed{\mathsf{R}\,\alpha}\to\boxed{\mathsf{R}\,\beta} \qquad (\ogreaterthan_{\mathsf{S}}) :: \boxed{\mathsf{S}\,\alpha\to\beta}\to\boxed{\mathsf{S}\,\alpha}\to\boxed{\mathsf{S}\,\beta}$$

$$F \ogreaterthan_{\mathsf{R}} A \coloneqq \lambda i.\, F\, i\, (A\, i) \qquad F \ogreaterthan_{\mathsf{S}} A \coloneqq \{f\, a \mid f \in F,\ a \in A\}$$

$$(\olessthan_{\mathsf{R}}) :: \boxed{\mathsf{R}\,\alpha}\to\boxed{\mathsf{R}\,\alpha\to\beta}\to\boxed{\mathsf{R}\,\beta} \qquad (\olessthan_{\mathsf{S}}) :: \boxed{\mathsf{S}\,\alpha}\to\boxed{\mathsf{S}\,\alpha\to\beta}\to\boxed{\mathsf{S}\,\beta}$$

$$A \olessthan_{\mathsf{R}} F \coloneqq \lambda i.\, F\, i\, (A\, i) \qquad A \olessthan_{\mathsf{S}} F \coloneqq \{f\, a \mid f \in F,\ a \in A\}$$

(28)



- But B&C prefer not to generalize to the worst case: "The rigid, pervasive replacement of ordinary combinatory modes with effect-specific variants limits the applicability of the fragment to just the specific effects described. What is gained in uniformity and simplicity is sacrificed in generality and extensibility."

- For example: "Simply mixing the two kinds of phenomena is beyond the reach of either grammar."
- Fortunately, it turns out that the Hamblin and H&K composition operations are special cases of the operations of a more general structure called an **applicative functor**. A functor is applicative if there exist η and $\circledast$ functions (pronounced "unit" and "apply") that satisfy the following laws:

(29) Applicative functor laws

Homomorphism

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

Identity

$$\eta\, \mathbf{id} \circledast X = X$$

Interchange

$$\eta\,(\lambda k.\, k\, x) \circledast F = F \circledast \eta\, x$$

Composition

$$(\eta\,(\circ) \circledast F \circledast G) \circledast X = F \circledast (G \circledast X)$$

- The η and $\circledast$ functions for the R and S functors are just the "lift" operations – for incorporating trivially effectful constituents – and the unidirectional versions of the combinators above. We are invited to confirm that they respect the laws in (29).

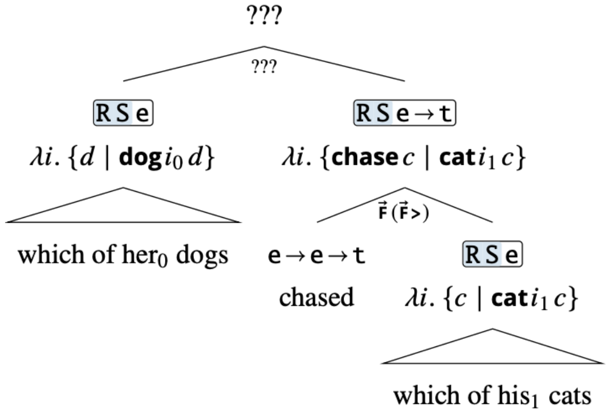
(30)

$$\eta_{\mathsf{R}}\, x \coloneqq \lambda i.\, x \qquad \eta_{\mathsf{S}}\, x \coloneqq \{x\}$$

$$F \circledast_{\mathsf{R}} X \coloneqq \lambda i.\, F\, i\, (X\, i) \qquad F \circledast_{\mathsf{S}} X \coloneqq \{f\, x \mid f \in F,\ x \in X\}$$

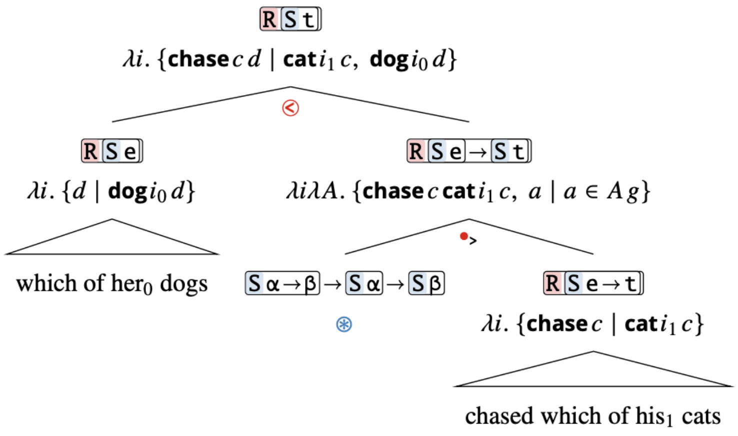
- Each functor that we paired with the phenomena above also gives rise to an applicative. So we could add η and $\circledast$ operations to the grammar for an arbitrary applicative functor Σ and approximate the Hamblin and H&K grammars.
- But we face a problem that parallels our difficulties with regular functors: siblings with nested effects.

(31)



- Just as in the previous chapter, the strategy will be to map the new mode of combination below outer effects.
- And again, one way of accomplishing this is to let $\circledast$ act as a covert operator.

(32)



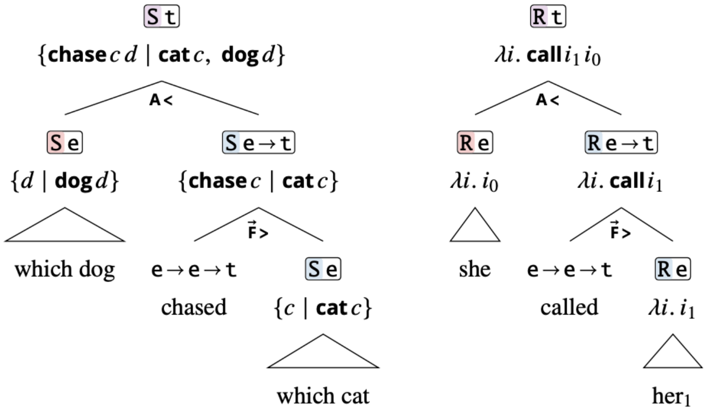
- And another is to introduce a higher order mode of combination. Whenever there is a mode $(\circledast)$ that might combine constituents $M :: \sigma$ and $N :: \tau$, then there should also be a mode to combine constituents $M' :: \Sigma\sigma$ and $N' :: \Sigma\tau$, so long as Σ is an applicative functor.

(33) Higher order applicative composition

$$\mathbf{A} :: (\sigma \to \tau \to \omega) \to \boxed{\Sigma\,\sigma} \to \boxed{\Sigma\,\tau} \to \boxed{\Sigma\,\omega}$$
$$\mathbf{A}\,(*)\,E_1\,E_2 \coloneqq \eta\,(*) \circledast E_1 \circledast E_2$$

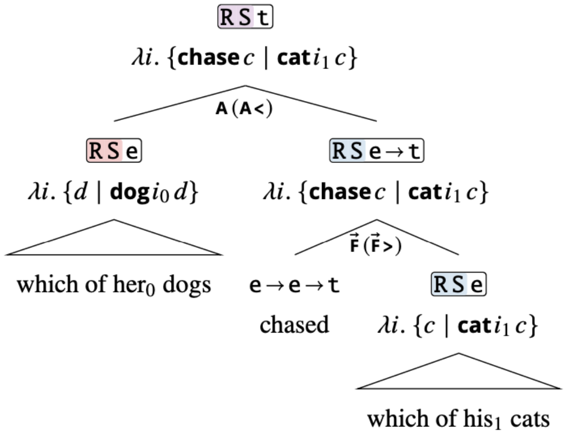
- The generalized pointwise and assignment sensitive modes of composition can be constructed with **A**.

(34) Higher order applicative composition



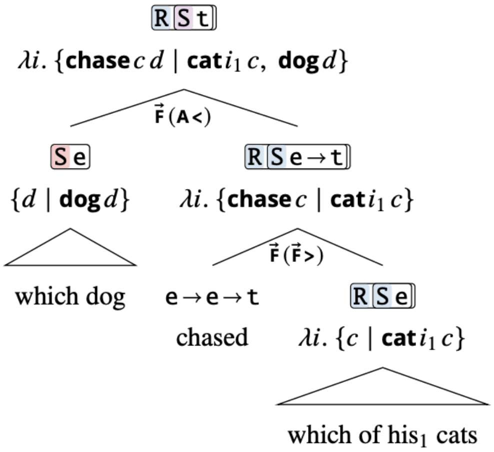
- To compose nested applicative functors, we let **A** apply twice.

(35)

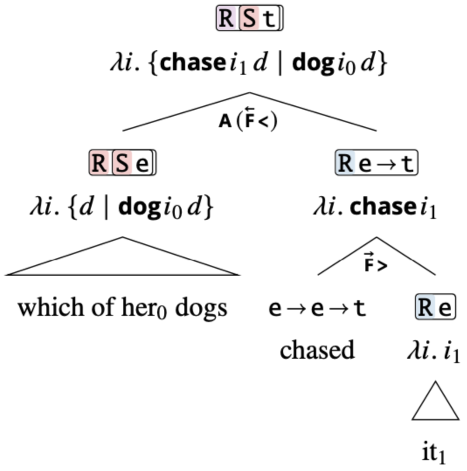


- Having $\overrightarrow{F}$, $\overleftarrow{F}$, and **A** in the grammar provides flexibility. Outer effects may be merged with **A** while inner ones are mapped over, or vice versa.

(36)



(37)



3.2 Selectivity & eliminating effects

- So far, the grammar offers no way to keep effects from bubbling to the top of a derivation. But there's empirical evidence for closure operators, either lexical or covert. In this framework, these items are functions that **eliminate effects**.

(38) Lexical closure operators

only :: $\boxed{\mathtt{F\,t}} \to \mathtt{t}$

$[\![\text{only}]\!] \coloneqq \lambda\langle q, \mathcal{A}\rangle.\, \{p \in \mathcal{A} \mid p\} = \{q\}$

mo :: $\boxed{\mathtt{S\,t}} \to \mathtt{t}$

$[\![\text{mo}]\!] \coloneqq \lambda m.\, \bigwedge m$

may :: $\boxed{\mathtt{S\,t}} \to \mathtt{t}$

$[\![\text{may}]\!] \coloneqq \lambda m.\, \bigwedge\{\Diamond p \mid p \in m\}$

wonder :: $\boxed{\mathtt{S\,t}} \to \mathtt{e} \to \mathtt{t}$

$[\![\text{wonder}]\!] \coloneqq \lambda m \lambda x.\, \textbf{wonder}\, m\, x$

and :: $\boxed{\mathtt{D\,t}} \to \boxed{\mathtt{D\,t}} \to \boxed{\mathtt{D\,t}}$

$[\![\text{and}]\!] \coloneqq \lambda R \lambda L \lambda i.\, \{\langle p \wedge q, k\rangle \mid \langle p, j\rangle \in L\,i,\ \langle q, k\rangle \in R\,j\}$

n :: $\boxed{\mathtt{V}\,\beta} \to \boxed{\mathtt{V\,e} \to \beta}$

$[\![n]\!] \coloneqq \lambda b \lambda g \lambda x.\, b\, g^{n \mapsto x}$

(39) Covert closure operators

∃-clo :: $\boxed{\mathtt{S\,t}} \to \mathtt{t}$

$[\![\exists\text{-clo}]\!] \coloneqq \lambda m.\, \bigvee m$

ScalAssert :: $\boxed{\mathtt{F\,t}} \to \mathtt{t}$

$[\![\text{ScalAssert}]\!] \coloneqq \lambda\langle q, \mathcal{A}\rangle.\, \{p \in \mathcal{A} \mid p \Rightarrow q\} = \{q\}$

$\Downarrow$ :: $\boxed{\mathtt{C\,t}} \to \mathtt{t}$

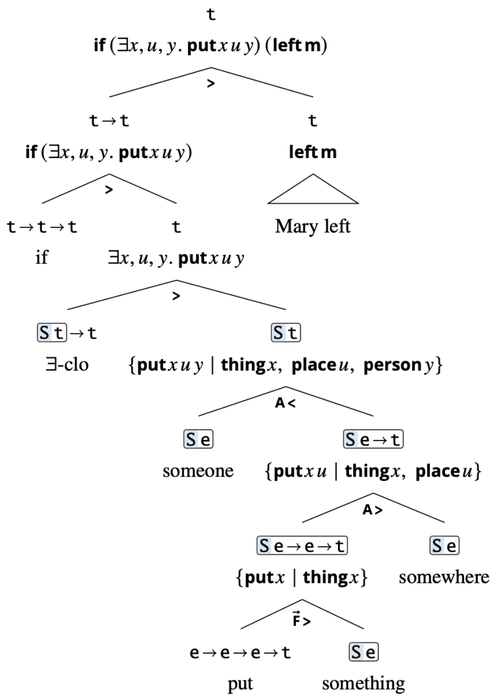
$[\![\Downarrow]\!] \coloneqq \lambda m.\, m\,(\lambda p.\, p)$

Accom :: $\boxed{\mathtt{M\,t}} \to \mathtt{t}$

$[\![\text{Accom}]\!] \coloneqq \lambda m.\, \textbf{false} \text{ if } m = \# \text{ else } m$

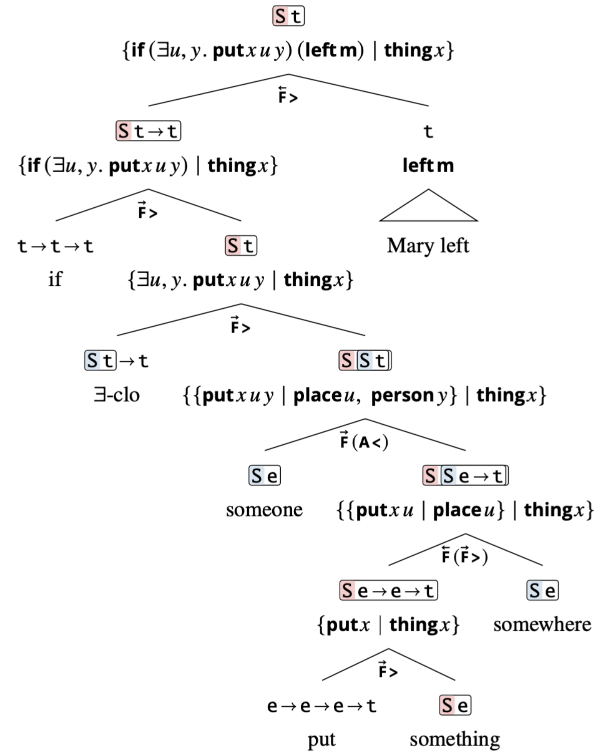
- Because applicative composition collapses effects, a closure operator that acts on the result of applicative composition will close over every effect in its scope, **unselectively**.

(40)



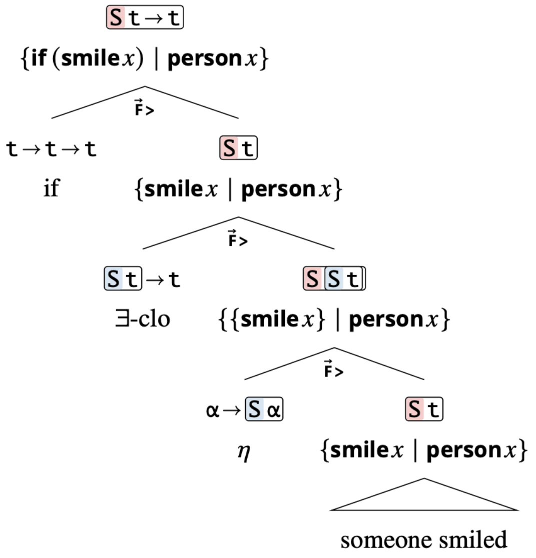
- But it's still possible to eliminate effects selectively, and to e.g. allow a single effect to bubble to the top of a derivation, by using an applicative functor's • operation rather than its ⊛ operation.
- The derivation below mimics the derivation above except that the alternatives generated by the direct object are consistently mapped-over at every node. The alternatives generated by the other two indefinites are merged as above and jointly closed over in the antecedent.

(41)


S t
{if (∃u, y. put x u y) (left m) | thing x}
F⃗>
S t→t
{if (∃u, y. put x u y) | thing x}
t
left m
F⃗>
t→t→t
if
S t
{∃u, y. put x u y | thing x}
Mary left
F⃗>
S t→t
∃-clo
S S t
{{put x u y | place u, person y} | thing x}
F⃗ (A<)
S e
someone
S S e→t
{{put x u | place u} | thing x}
F⃗ (F⃗>)
S e→e→t
{put x | thing x}
S e
somewhere
F⃗>
e→e→e→t
put
S e
something

- Now here is a problem: with η freely applying in the grammar, we predict that effects can always escape closure operators. This will happen if η applies to an operator's complement, causing the operator to only eliminate the trivial effect that η introduces.

(42)



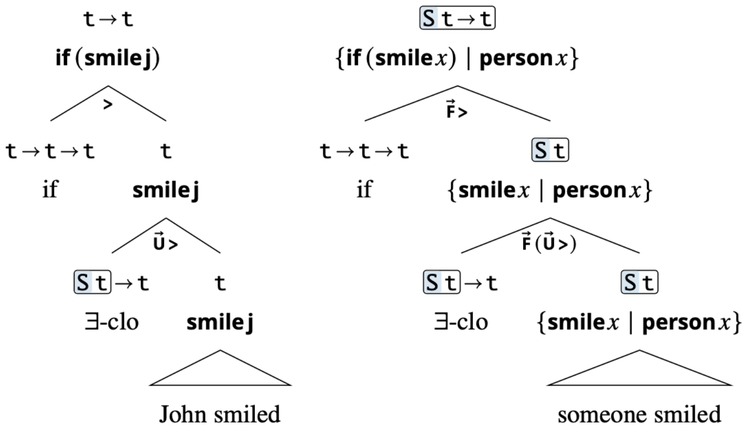
- The solution that B&C propose is to not let η freely apply, but rather to use it only in the course of applying a closure operator to a complement that lacks effects: "the only semantic use for η is to feed a closure operator of some sort or another".

(43) Closure composition

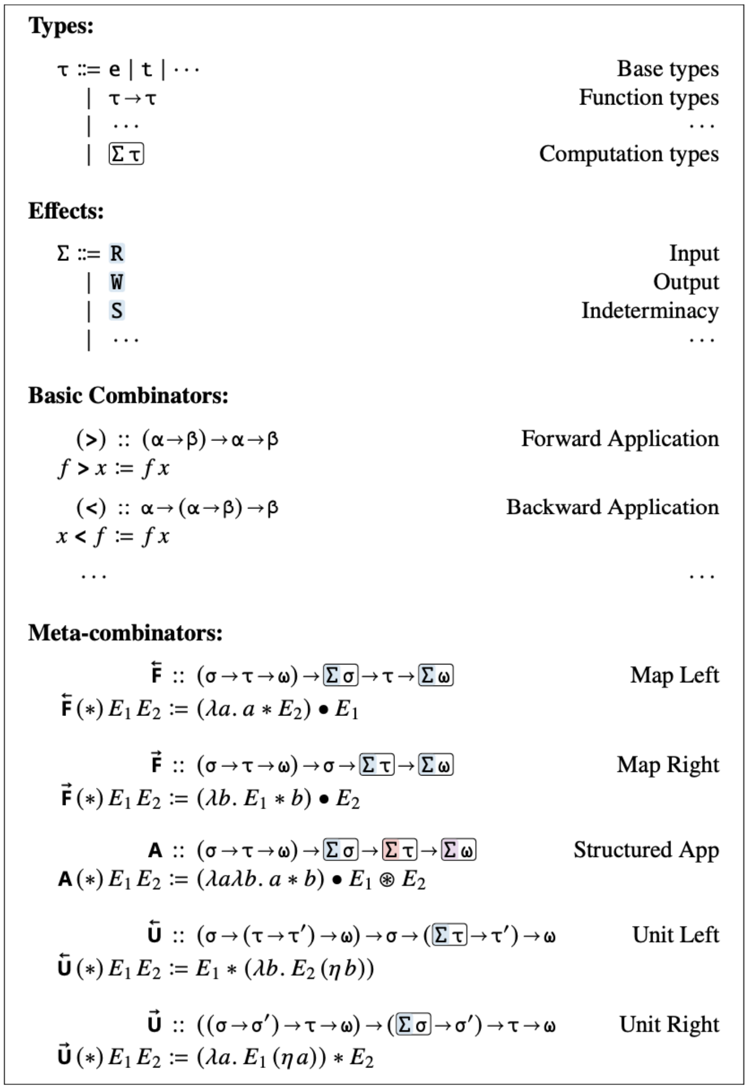
$$\vec{\mathbf{U}} :: ((\sigma \to \sigma') \to \tau \to \omega) \to (\boxed{\Sigma\,\sigma} \to \sigma') \to \tau \to \omega$$
$$\vec{\mathbf{U}}\,(*)\,E_1\,E_2 \coloneqq (\lambda a.\, E_1\,(\eta\,a)) * E_2$$
$$\overleftarrow{\mathbf{U}} :: (\tau \to (\sigma \to \sigma') \to \omega) \to \tau \to (\boxed{\Sigma\,\sigma} \to \sigma') \to \omega$$
$$\overleftarrow{\mathbf{U}}\,(*)\,E_1\,E_2 \coloneqq E_1 * (\lambda b.\, E_2\,(\eta\,b))$$

- We use **U** like so:

(44)



- Here is the grammar so far. "It's in some sense maximally expressive relative to the applicative structure of an effect. It permits every possible agglomeration or stratification of structure, which in turn means that closure operators may capture anywhere from none to all of the effects in their prejacents."



Types:

$\tau ::= \mathsf{e} \mid \mathsf{t} \mid \cdots$ Base types

$\quad\mid\ \tau \to \tau$ Function types

$\quad\mid\ \cdots$ $\cdots$

$\quad\mid\ \boxed{\Sigma\,\tau}$ Computation types

Effects:

$\Sigma ::= \mathtt{R}$ Input

$\quad\mid\ \mathtt{W}$ Output

$\quad\mid\ \mathtt{S}$ Indeterminacy

$\quad\mid\ \cdots$ $\cdots$

Basic Combinators:

$(\mathbf{>}) :: (\alpha \to \beta) \to \alpha \to \beta$ Forward Application

$f \mathbin{\mathbf{>}} x \coloneqq f\,x$

$(\mathbf{<}) :: \alpha \to (\alpha \to \beta) \to \beta$ Backward Application

$x \mathbin{\mathbf{<}} f \coloneqq f\,x$

$\cdots$ $\cdots$

Meta-combinators:

$\overleftarrow{\mathbf{F}} :: (\sigma \to \tau \to \omega) \to \boxed{\Sigma\,\sigma} \to \tau \to \boxed{\Sigma\,\omega}$ Map Left

$\overleftarrow{\mathbf{F}}\,(*)\,E_1\,E_2 \coloneqq (\lambda a.\, a * E_2) \bullet E_1$

$\overrightarrow{\mathbf{F}} :: (\sigma \to \tau \to \omega) \to \sigma \to \boxed{\Sigma\,\tau} \to \boxed{\Sigma\,\omega}$ Map Right

$\overrightarrow{\mathbf{F}}\,(*)\,E_1\,E_2 \coloneqq (\lambda b.\, E_1 * b) \bullet E_2$

$\mathbf{A} :: (\sigma \to \tau \to \omega) \to \boxed{\Sigma\,\sigma} \to \boxed{\Sigma\,\tau} \to \boxed{\Sigma\,\omega}$ Structured App

$\mathbf{A}\,(*)\,E_1\,E_2 \coloneqq (\lambda a \lambda b.\, a * b) \bullet E_1 \circledast E_2$

$\overleftarrow{\mathbf{U}} :: (\sigma \to (\tau \to \tau') \to \omega) \to \sigma \to (\boxed{\Sigma\,\tau} \to \tau') \to \omega$ Unit Left

$\overleftarrow{\mathbf{U}}\,(*)\,E_1\,E_2 \coloneqq E_1 * (\lambda b.\, E_2\,(\eta\,b))$

$\overrightarrow{\mathbf{U}} :: ((\sigma \to \sigma') \to \tau \to \omega) \to (\boxed{\Sigma\,\sigma} \to \sigma') \to \tau \to \omega$ Unit Right

$\overrightarrow{\mathbf{U}}\,(*)\,E_1\,E_2 \coloneqq (\lambda a.\, E_1\,(\eta\,a)) * E_2$

- We can consider how the grammar behaves when deprived of particular operators.
  - With only $\bullet$, closure operators will need to capture the effect of one and only one constituent.
  - With only $\circledast$, every constituent must be effectful, except closure operators, and closure operators will capture every effect in their scope.

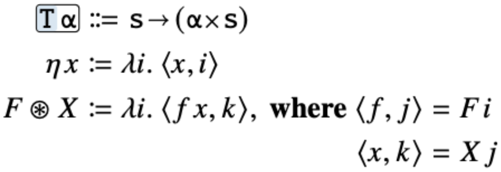
3.3 Commutativity

- The R and S functors are commutative, in the following sense. For any $F :: \Sigma e \to t$ and $X :: \Sigma e$,

$$\mathbf{A} > F X = \mathbf{A} > X F$$

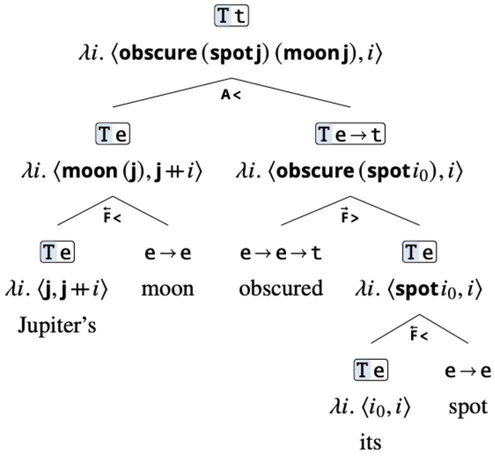
- The T and C functors are not commutative.

(45) State updating functor



$$\boxed{\mathsf{T}\,\alpha} ::= \mathsf{s} \to (\alpha \times \mathsf{s})$$
$$\eta\,x \coloneqq \lambda i.\, \langle x, i \rangle$$
$$F \circledast X \coloneqq \lambda i.\, \langle f\,x, k \rangle,\ \textbf{where}\ \langle f, j \rangle = F\,i$$
$$\langle x, k \rangle = X\,j$$

(46) Anaphora with the T functor

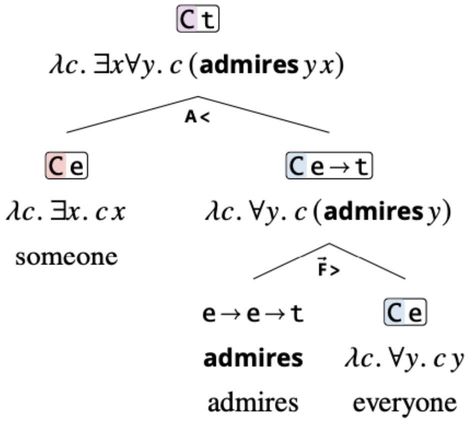


$\boxed{T\,t}$
$\lambda i.\, \langle \mathbf{obscure}\,(\mathbf{spot}\,\mathbf{j})\,(\mathbf{moon}\,\mathbf{j}), i\rangle$
A<
$\boxed{T\,e}$
$\lambda i.\, \langle \mathbf{moon}\,(\mathbf{j}), \mathbf{j} \mathbin{+\!\!+} i\rangle$
$\boxed{T\,e \to t}$
$\lambda i.\, \langle \mathbf{obscure}\,(\mathbf{spot}\,i_0), i\rangle$
$\overleftarrow{F}$<
$\boxed{T\,e}$
$\lambda i.\, \langle \mathbf{j}, \mathbf{j} \mathbin{+\!\!+} i\rangle$
Jupiter's
$e \to e$
moon
$\overrightarrow{F}$>
$e \to e \to t$
obscured
$\boxed{T\,e}$
$\lambda i.\, \langle \mathbf{spot}\,i_0, i\rangle$
$\overleftarrow{F}$<
$\boxed{T\,e}$
$\lambda i.\, \langle i_0, i\rangle$
its
$e \to e$
spot

(47) Continuation functor

$$\boxed{C\,\alpha} ::= (\alpha \to t) \to t$$
$$\eta\, x := \lambda c.\, c\, x$$
$$F \circledast X := \lambda c.\, F\,(\lambda f.\, X\,(\lambda x.\, c\,(f\,x)))$$

(48) Quantification with the C functor



$\boxed{C\,t}$
$\lambda c.\, \exists x \forall y.\, c\,(\mathbf{admires}\,y\,x)$
A<
$\boxed{C\,e}$
$\lambda c.\, \exists x.\, c\,x$
someone
$\boxed{C\,e \to t}$
$\lambda c.\, \forall y.\, c\,(\mathbf{admires}\,y)$
$\overrightarrow{F}$>
$e \to e \to t$
admires
admires
$\boxed{C\,e}$
$\lambda c.\, \forall y.\, c\,y$
everyone

4 Monads

4.1 Motivation

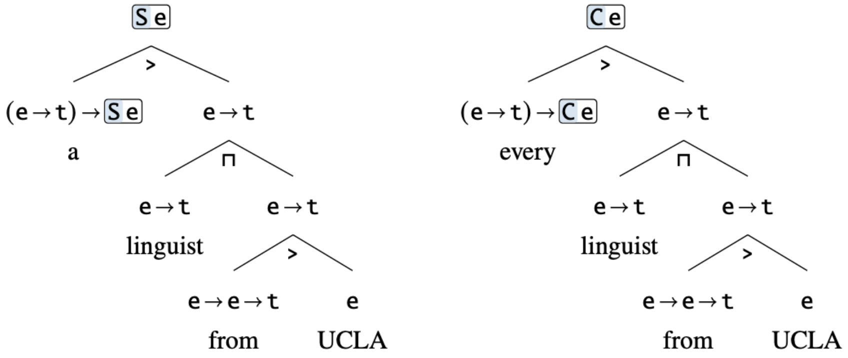
- Just as there are closure operators that *eliminate* effects, there are counterpart operations that *introduce* effects. Operators with these types are known as Kleisli arrows, while closure operators are known as co-Kleisli arrows.

(49) Determiners in the guise of Kleisli arrows

Expression	Type	Denotation
a	$(\mathtt{e}\to\mathtt{t})\to\boxed{\mathtt{S\,e}}$	$\lambda P.\,\{x \mid P\,x\}$
the	$(\mathtt{e}\to\mathtt{t})\to\boxed{\mathtt{M\,e}}$	$\lambda P.\,x$ if $P = \{x\}$ else #
no	$(\mathtt{e}\to\mathtt{t})\to\boxed{\mathtt{C\,e}}$	$\lambda P \lambda Q.\,\neg\exists x.\,P\,x \wedge Q\,x$
...	$(\mathtt{e}\to\mathtt{t})\to\boxed{\Sigma\,\mathtt{e}}$	...

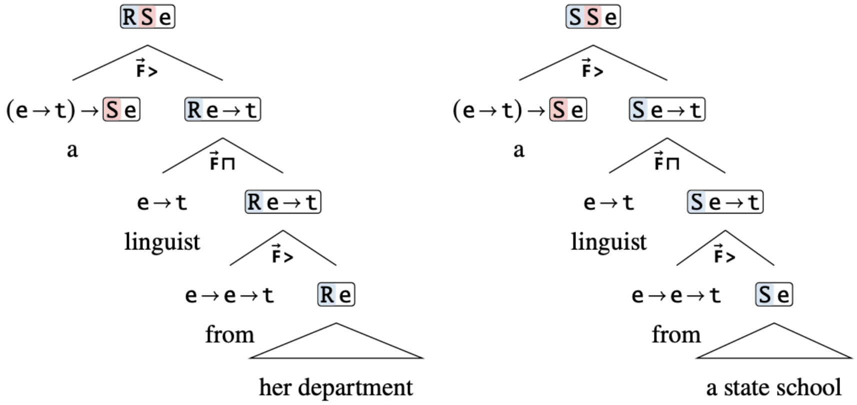
- The grammar so far is able to compose these operators.

(50) DP Internal Composition



- Up to a point, it can also compose doubly effectful determiners, for instance determiners whose restrictions are themselves effectful.

(51) Doubly Effectful DP Composition

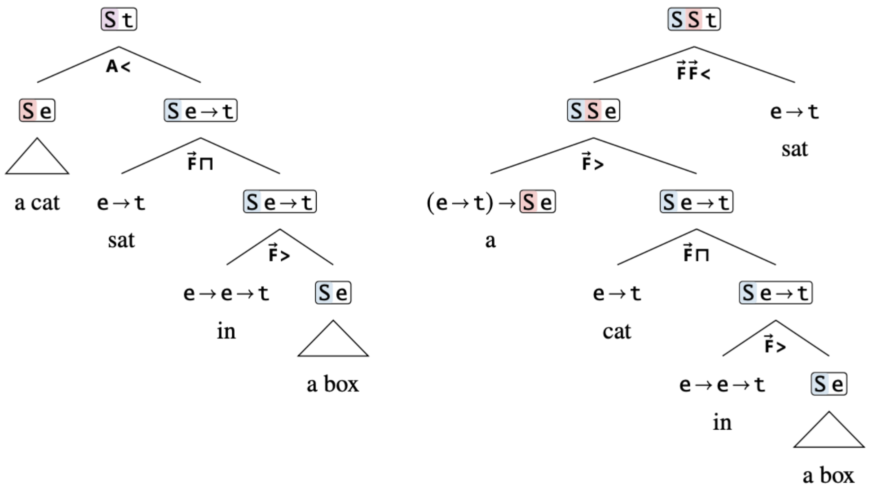


- But whereas applicative functors allowed us to collapse higher order effects into singly layered effects, Kleisli arrows appear to introduce stubbornly higher order effects.

- For the *RS* DP in (51), this is unsurprising, since the effects are different. But it's less clear why we shouldn't be able to reduce the *SS* DP into a simple *S* type, representing the set of all linguists from state schools. Nevertheless, it's just a fact about the grammar so far: there's no operation that will do it for us.

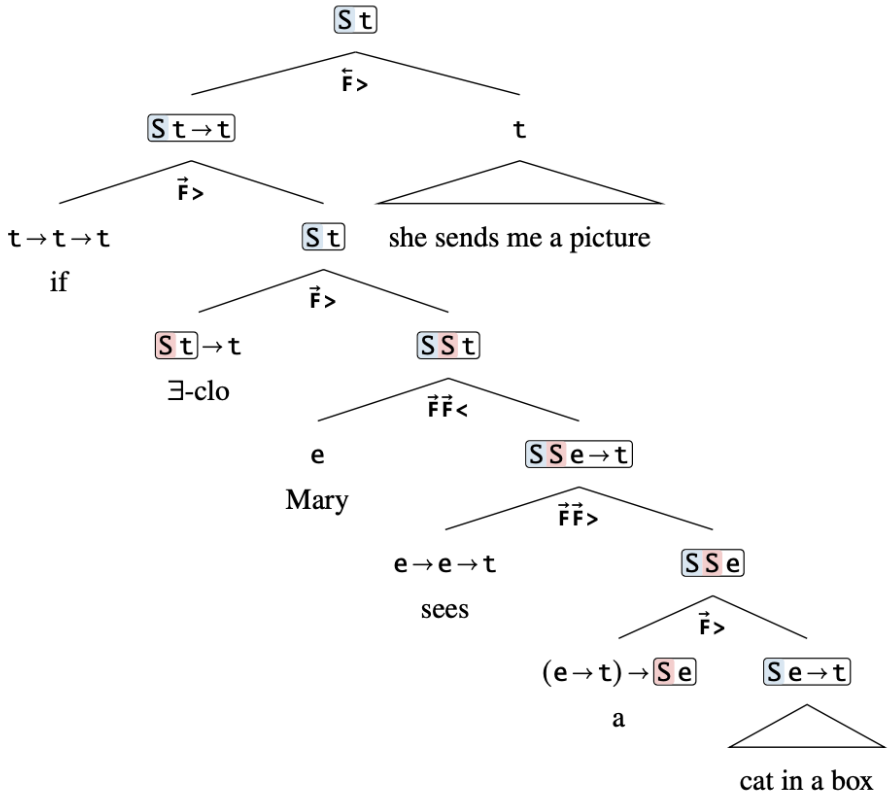
- It is illustrative to directly compare a derivation in which the applicative structure is collapsible to one in which it isn't. We see that the culprit is the Kleisli arrow.

(52) Applicative vs Kleisli structures



- “This predicts that an otherwise unselective closure operator will, extraor- dinarily, fail to capture effects that happen to be nested in the arguments of Kleisli arrows. For instance, assuming as in Chapter 3 that the antecedent of a conditional is associated with an existential closure operator, the current state of affairs predicts that a nested indefinite will necessarily take exceptional scope over the conditional.”

(53)



- We can construct somewhat parallel examples with free choice introducing and eliminating operators, e.g. “can” and “any”.

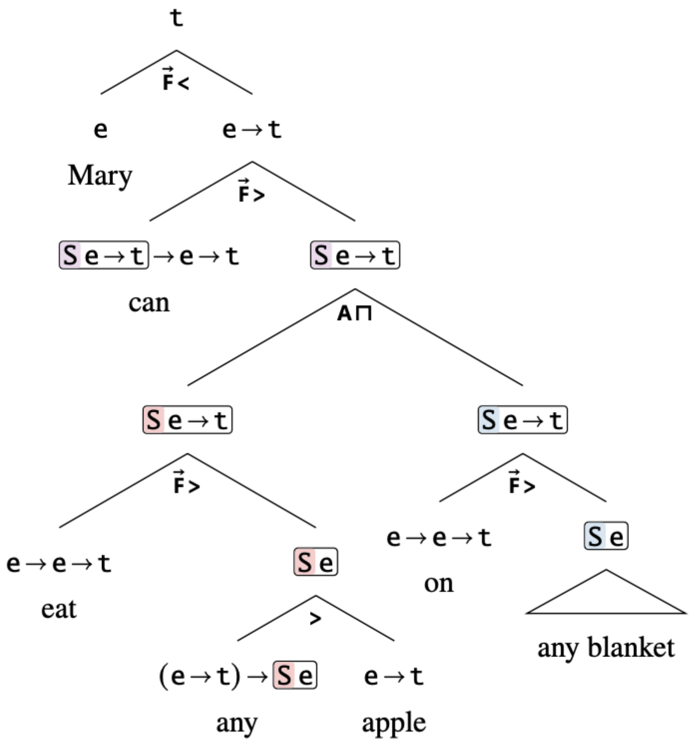
(54)

$$\text{can} :: \boxed{\mathsf{S}\ \mathsf{e} \to \mathsf{t}} \to \mathsf{e} \to \mathsf{t}$$

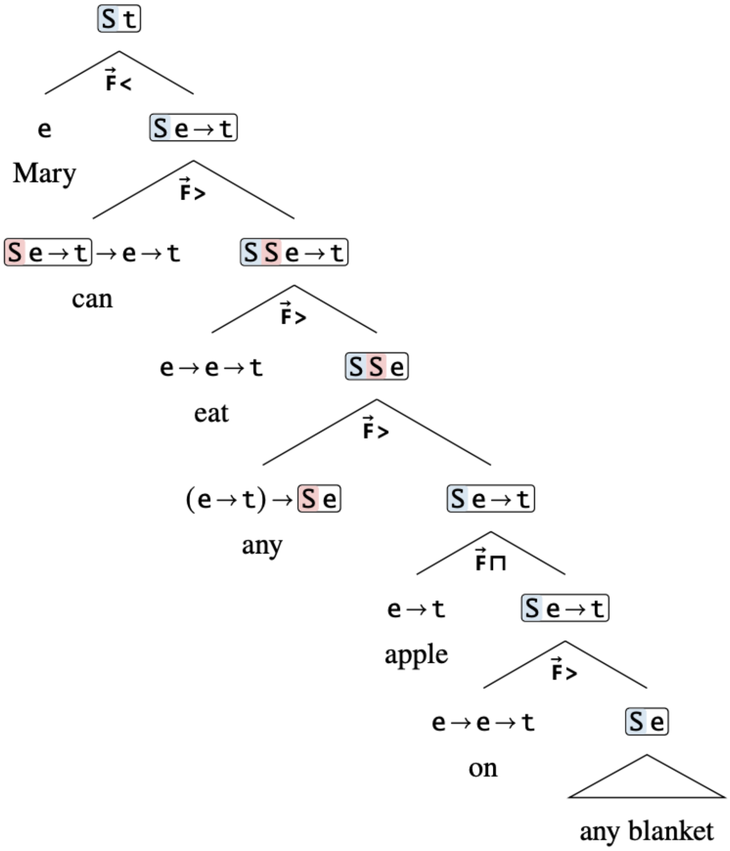
$$[\![\text{can}]\!] \coloneqq \lambda m \lambda x.\ \bigwedge\{\Diamond(P x) \mid P \in m\}$$

- For the sentence “Mary can eat any apple on any blanket”, we’re only able to flatten effects that are introduced with a particular PP attachment site.

(55)



(56)



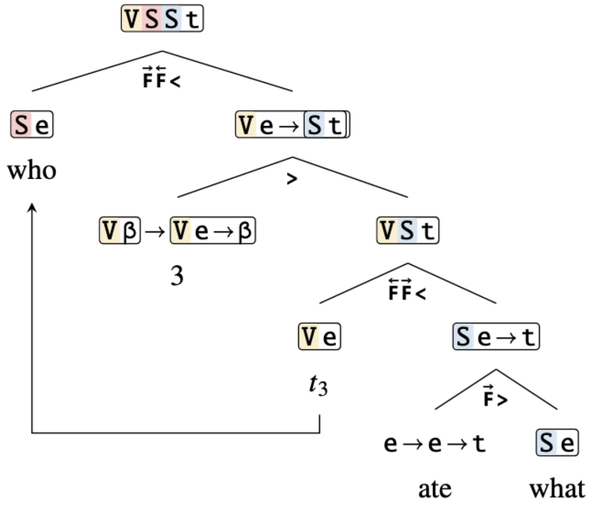
- The H&K abstraction operator will behave like a Kleisli arrow when its complement contains an effect.

(57)

$$n :: \boxed{V\,\beta} \to \boxed{V\,e \to \beta}$$
$$[\![n]\!] \coloneqq \lambda b \lambda g \lambda x.\, b\, g^{n\mapsto x}$$

- In (58), for instance, the inner *SS* cannot be reduced to a set of pairs of people and things in the eating relation.

(58)



References

Hamblin, C. L. (1976). Questions in montague english.

Heim, I. and A. Kratzer (1998). *Semantics in Generative Grammar*. Malden, MA: Blackwell.