# Effects as Sessions, Sessions as Effects

**Dominic Orchard**[1,2], Nobuko Yoshida[1]

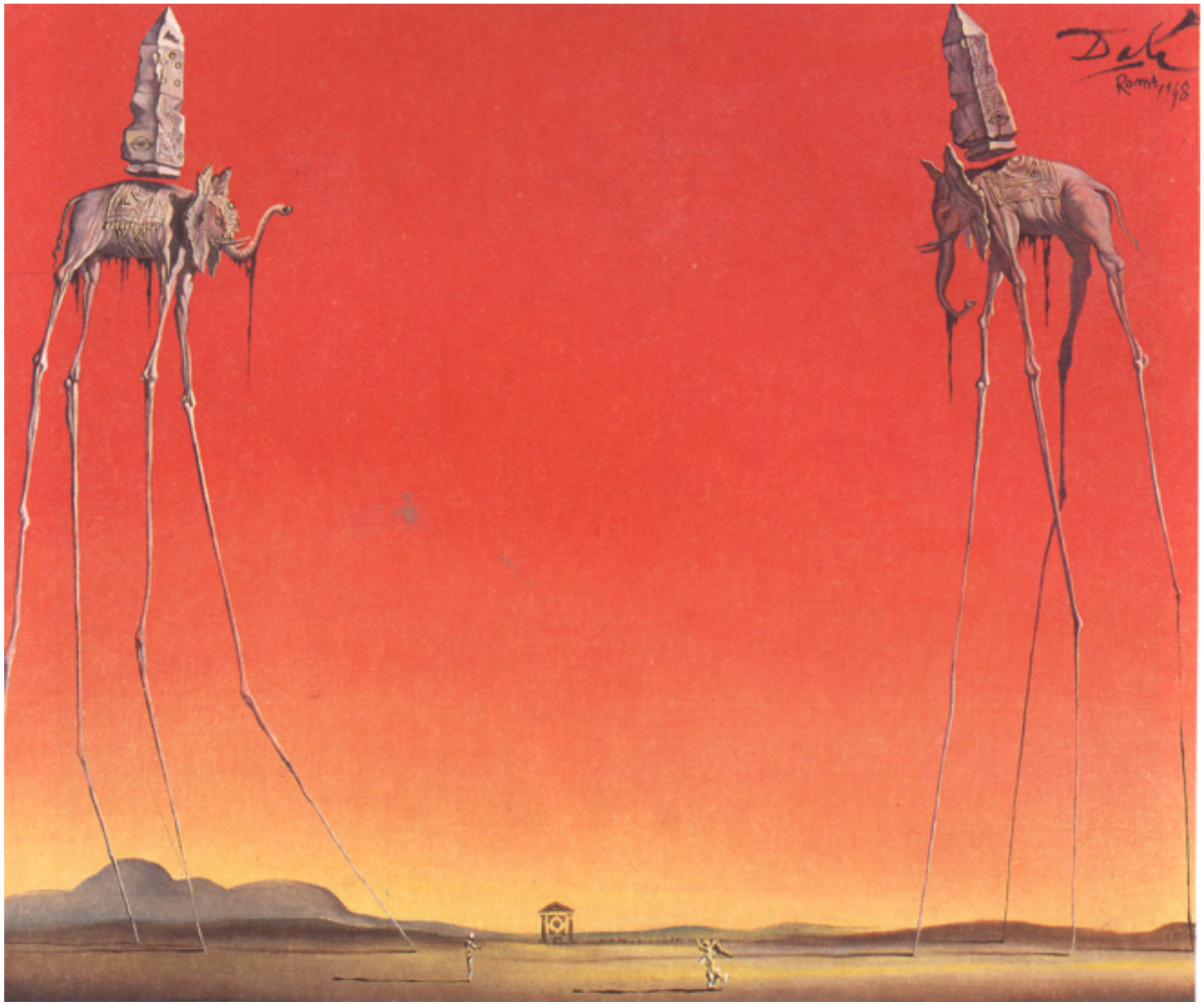
[1] Imperial College London

[2] University of Cambridge

dorchard.co.uk

λ-calculus

π-calculus




Dali's "Los Elefantes"

***parallel, effectful PCF**

λ-calculus

*Functions as processes*, Milner (1992)

*Milner (1980)* CCS

∩| ↓

π-calculus

Church (1930s)

*A Calculus of Mobile Processes, Milner, Parrow, Walker* (1992)

∪| ↑

*Functions as session-typed processes*,
Toninho, Caires, Pfenning (2012)

∪| ↑

simple types

session types

Church (1940)

\+ |

**effect systems**

Gifford, Lucassen (1986)

*Honda, Vasconcelos, Kubo* (1998)
*Yoshida, Vasconcelos* (2007)

this work

*Higher-Order Concurrent Programs with Finite Communication Topology.* Nielson, Nielson (1994)

$\Gamma \vdash M : \tau,\ \mathsf{F}$

$\Gamma\ ;\ \Delta \vdash P$

PCF - type-and-effect system

$(\mathsf{F}, \bullet, \mathrm{I}, \oplus, {}^*, \&, \sqsubseteq)$

π-calculus - session types

**value environment**

$x_1 : A_1,\ \ldots,\ x_n : A_n$

**session environment**

$c_1 : S_1,\ \ldots,\ c_n : S_n$

# π-calculus recap

| | |
|---|---|
| $c!\langle V\rangle.P$ | send $V$ on $c$ then $P$ |
| $c?(x).P$ | receive on $c$, bind to $x$, then |
| $P \mid Q$ | parallel |
| $\boldsymbol{\nu} c\ (P)$ | restriction (create channel) |
| $*c?(x).P$ | replicated input |
| $\mathbf{0}$ | inactive process |

**dual end-point**

$$(c?(x).P \mid \overline{c}!\langle V\rangle.Q) \rightarrow (P[V/x] \mid Q) \qquad (\beta \text{ reduction})$$

$$(*c?(x).P \mid \overline{c}!\langle V\rangle.Q) \rightarrow (*c?(x).P \mid P[V/x] \mid Q) \qquad (*\beta \text{ reduction})$$

# π-calculus recap (2)

$c \triangleright \{L_1 : P_1, \ldots, L_n : P_n\}$ offer *n* labelled choices

$c \triangleleft L_i . P$ select label *i* then act P

$(c \triangleright \{L_1 : P_1, \ldots, L_n : P_n\} \mid \overline{c} \triangleleft L_i . Q) \rightarrow (P_i \mid Q)$

(β reduction)

(commutativity)

$P \mid Q \equiv Q \mid P$

(associativity)

$P \mid (Q \mid R) \equiv (P \mid Q) \mid R$

(scope extrusion)

$\boldsymbol{\nu} c\ (P \mid Q) \equiv \boldsymbol{\nu} c\ (P) \mid Q \quad (if\ \mathrm{c} \# Q)$

# Session type primer

$$S ::= ![\tau].S \;\mid\; ?[\tau].S \;\mid\; \ldots$$

send a $\tau$ recv a $\tau$

e.g. $d : ?[![\mathsf{bool}]],\ c : ![\mathsf{int}] \vdash c!\langle 3\rangle.d?(e).e!\langle \mathrm{true}\rangle$

# Session type primer

$$S ::= ![\tau].S \;\mid\; ?[\tau].S \;\mid\; \&[l_1 : S_1 \ldots l_n : S_n] \;\mid\; \oplus[l_1 : S_1 \ldots l_n : S_n]$$

send a $\tau$ recv a $\tau$ offer choice 1-n select choice 1-n

e.g. $d : ?[![\text{bool}]],\ c : ![\text{int}] \vdash c!\langle 3\rangle.d?(e).e!\langle \text{true}\rangle$

# Session type primer

$$S ::= ![\tau].S \mid ?[\tau].S \mid \&[l_1 : S_1 \ldots l_n : S_n] \mid \oplus[l_1 : S_1 \ldots l_n : S_n]$$

send a $\tau$ | recv a $\tau$ | offer choice 1-n | select choice 1-n

Duality: *ensures absence of communication errors*

$$dual(?[\tau].S) = ![\tau].dual(S)$$ send is dual to receive

$$dual(\&[l_1 : S_1 \ldots l_n : S_n]) = \oplus[l_1 : dual(S_1) \ldots l_n : dual(S_n)]$$

'offer' dual to 'select'

$$\frac{\Delta,\ c : S,\ \overline{c} : dual(S) \vdash P}{\Delta \vdash \boldsymbol{\nu} c.P} \qquad \frac{\Gamma\ ;\ \Delta_1 \vdash P \qquad \Gamma\ ;\ \Delta_2 \vdash Q}{\Gamma\ ;\ \Delta_1 \odot \Delta_2 \vdash P \mid Q}$$

# Session type primer

$$S ::= ![\tau].S \mid ?[\tau].S \mid \&[l_1 : S_1 \ldots l_n : S_n] \mid \oplus[l_1 : S_1 \ldots l_n : S_n]$$

send a $\tau$ | recv a $\tau$ | offer choice 1-n | select choice 1-n

Duality: *ensures absence of communication errors*

$$dual(?[\tau].S) = ![\tau].dual(S)$$ send is dual to receive

$$dual(\&[l_1 : S_1 \ldots l_n : S_n]) = \oplus[l_1 : dual(S_1) \ldots l_n : dual(S_n)]$$

'offer' dual to 'select'

$$\frac{c : ![\mathsf{bool}] \vdash P \qquad \overline{c} : ?[\mathsf{int}] \vdash Q}{(c : ![\mathsf{bool}]) \odot (\overline{c} : ?[\mathsf{int}]) \vdash P \mid Q}$$

# Session type primer

$$S ::= ![\tau].S \mid ?[\tau].S \mid \&[l_1 : S_1 \ldots l_n : S_n] \mid \oplus[l_1 : S_1 \ldots l_n : S_n]$$

send a $\tau$ | recv a $\tau$ | offer choice 1-n | select choice 1-n

Duality: *ensures absence of communication errors*

$$dual(?[\tau].S) = ![\tau].dual(S)$$ send is dual to receive

$$dual(\&[l_1 : S_1 \ldots l_n : S_n]) = \oplus[l_1 : dual(S_1) \ldots l_n : dual(S_n)]$$

'offer' dual to 'select'

$$\frac{c : ![\mathsf{int}].![\mathsf{int}] \vdash P \qquad \overline{c} : ?[\mathsf{int}] \vdash Q}{(c : ![\mathsf{int}].![\mathsf{int}]) \odot (\overline{c} : ?[\mathsf{int}]) \vdash P \mid Q}$$

# Session type primer

$$S ::= ![\tau].S \mid ?[\tau].S \mid \&[l_1 : S_1 \ldots l_n : S_n] \mid \oplus[l_1 : S_1 \ldots l_n : S_n]$$

send a $\tau$ | recv a $\tau$ | offer choice 1-n | select choice 1-n

Duality: *ensures absence of communication errors*

$$dual(?[\tau].S) = ![\tau].dual(S)$$ send is dual to receive

$$dual(\&[l_1 : S_1 \ldots l_n : S_n]) = \oplus[l_1 : dual(S_1) \ldots l_n : dual(S_n)]$$

'offer' dual to 'select'

$$\frac{c : ![\mathsf{int}] \vdash P \qquad \overline{c} : ?[\mathsf{int}] \vdash Q}{(c : ![\mathsf{int}]) \odot (\overline{c} : ?[\mathsf{int}]) \vdash P \mid Q}$$

✔

# Session type primer

$$S ::= ![\tau].S \mid ?[\tau].S \mid \&[l_1 : S_1 \ldots l_n : S_n] \mid \oplus[l_1 : S_1 \ldots l_n : S_n]$$

send a $\tau$ | recv a $\tau$ | offer choice 1-n | select choice 1-n

Duality: *ensures absence of communication errors*

$$dual(?[\tau].S) = ![\tau].dual(S)$$ send is dual to receive

$$dual(\&[l_1 : S_1 \ldots l_n : S_n]) = \oplus[l_1 : dual(S_1) \ldots l_n : dual(S_n)]$$

'offer' dual to 'select'

$$\frac{c : ![\mathsf{int}] \vdash P \qquad \overline{c} : ?[\mathsf{int}].?[\mathsf{int}] \vdash Q \qquad c : ![\mathsf{int}] \vdash R}{(c : ![\mathsf{int}]) \odot (\overline{c} : ?[\mathsf{int}].?[\mathsf{int}]) \odot (c : ![\mathsf{int}]) \vdash P \mid Q \mid R}$$

# Session type primer

$$S ::= ![\tau].S \mid ?[\tau].S \mid \&[l_1 : S_1 \ldots l_n : S_n] \mid \oplus[l_1 : S_1 \ldots l_n : S_n]$$

send a $\tau$ | recv a $\tau$ | offer choice 1-n | select choice 1-n

Duality: *ensures absence of communication errors*

$$dual(?[\tau].S) = ![\tau].dual(S)$$ send is dual to receive

$$dual(\&[l_1 : S_1 \ldots l_n : S_n]) = \oplus[l_1 : dual(S_1) \ldots l_n : dual(S_n)]$$

'offer' dual to 'select'

$$\frac{c : ![\mathsf{int}] \vdash P \qquad \overline{c} : ?[\mathsf{int}], \overline{d} : ?[\mathsf{int}] \vdash Q \qquad d : ![\mathsf{int}] \vdash R}{(c : ![\mathsf{int}], \overline{c} : ?[\mathsf{int}], d : ![\mathsf{int}], \overline{d} : ?[\mathsf{int}]) \vdash P \mid Q \mid R}$$ ✔

# Session type primer

$$S ::= ![\tau].S \mid ?[\tau].S \mid \&[l_1 : S_1 \ldots l_n : S_n] \mid \oplus[l_1 : S_1 \ldots l_n : S_n]$$
$$\mid \mu\alpha.S \mid \alpha \mid *![\tau].S \mid \mathbf{end}$$

Duality: *ensures absence of communication errors*

$$dual(?[\tau].S) = ![\tau].dual(S)$$ send is dual to receive

$$dual(\&[l_1 : S_1 \ldots l_n : S_n]) = \oplus[l_1 : dual(S_1) \ldots l_n : dual(S_n)]$$

'offer' dual to 'select'

# Session types as “linear” capabilities

$$\frac{c : ![\tau].S \vdash c?(x).P \qquad \overline{c} : ?[\tau].T \vdash \overline{c}!\langle V\rangle.Q}{c : ![\tau].S,\ \overline{c} : ?[\tau].T \vdash c?(x).P \mid \overline{c}!\langle V\rangle.Q}$$

$$\Big\downarrow \beta \qquad\qquad \Big\downarrow \beta$$

$$c : S,\ \overline{c} : T \vdash P[V/x] \mid Q$$

# PCF + effects (parameterised)

$\Gamma \vdash M : \tau, \mathsf{F}$

$$\text{abs}\ \frac{\Gamma, x : \sigma \vdash M : \tau, \mathsf{F}}{\Gamma \vdash \lambda x.M : \sigma \xrightarrow{\mathsf{F}} \tau, \mathrm{I}} \qquad \text{app}\ \frac{\Gamma \vdash M : \sigma \xrightarrow{\mathsf{H}} \tau, \mathsf{F} \qquad \Gamma \vdash N : \sigma, \mathsf{G}}{\Gamma \vdash M\ N : \tau, \mathsf{F} \bullet \mathsf{G} \bullet \mathsf{H}}$$

$$\text{var}\ \frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma, \mathrm{I}} \qquad \text{sub}\ \frac{\Gamma \vdash M : \tau, \mathsf{F} \qquad \mathsf{F} \sqsubseteq \mathsf{G}}{\Gamma \vdash M : \tau, \mathsf{G}}$$

$$\text{case}\ \frac{\Gamma \vdash M : \mathsf{nat}, \mathsf{F} \qquad \Gamma \vdash N_1 : \tau, \mathsf{G} \qquad \Gamma, x : \mathsf{nat} \vdash N_1 : \tau, \mathsf{H}}{\Gamma \vdash \mathbf{case}\ M\ \mathbf{of}\ 0 \mapsto N_1, (\mathsf{suc}\ x) \mapsto N_2 : \tau, \mathsf{F} \bullet (\mathsf{G} \oplus \mathsf{H})}$$

$$\text{rec}\ \frac{\Gamma, x : \tau, f : \tau \xrightarrow{\mathsf{F}} \tau, \vdash M : \tau, \mathsf{F}}{\Gamma \vdash \mathbf{rec}\ (\lambda f.\lambda x.M) : \tau \xrightarrow{\mathsf{F}} \tau, \mathrm{I}} \qquad \text{par}\ \frac{\Gamma \vdash M : \mathsf{unit}, \mathsf{F} \qquad \Gamma \vdash N : \mathsf{unit}, \mathsf{G}}{\Gamma \vdash M \mid N : \mathsf{unit}, \mathsf{F}\ \&\ \mathsf{G}}$$

- $(\mathsf{F}, \bullet, \mathrm{I})$ sequential composition
- $\sqsubseteq$ sub effecting (approximation)
- & parallell composition
- $\oplus$ alternation (**case**)
- $*$ recursion

# Effects vs. session types

| effect systems | | session types | |
|---|---|---|---|
| $\bullet$ | sequential | $?[\tau].\text{-}$ $\quad$ $![\tau].\text{-}$ | prefixing |
| $\mathrm{I}$ | pure | $\mathbf{end}$ | inaction |
| $\oplus$ | conditional | $\oplus[\mathrm{l}_1 : S_1, \ \ldots \ \mathrm{l}_\mathrm{n} : S_\mathrm{n}]$ | selection/ |
| | | $\&[\mathrm{l}_1 : S_1, \ \ldots \ \mathrm{l}_\mathrm{n} : S_\mathrm{n}]$ | branching |
| $*$ | repetition | $\mu a \ . \ S$ $\quad$ $a$ | fixed-points |
| | | $*![\tau]$ $\ *?[\tau]$ | replication |
| $\sqsubseteq$ | sub-effecting | $\leqq$ | sub-typing |
| $\&$ | parallelism | $\odot$ | balancing |

# Effects as sessions

$$[\![\ \Gamma \vdash M : \tau,\ \mathsf{F}\ ]\!]^{eff}_{r} \longrightarrow [\![ F\Gamma\ ]\!], r : ![\![\tau]\!],\ eff : [\![\ \mathsf{F}\ ]\!] \vdash$$

$r$ - channel which sends result

$eff$ - (**effect channel**) over which effects are simulated

idea: encode effect info. as session types

# Sessions as effects

$$\Gamma; \Delta \vdash P \longrightarrow [\![\ \Gamma\ ]\!] \vdash M : \mathbf{unit}, [\![\ \Delta\ ]\!]$$

idea: instantiate effect with "session effects"

# Parameters to the encoding (effects into sessions)


- **Effect-dependent parameters:**
  - ➡ *Effect handler*

    [Plotkin, Pretnar "Handlers of Algebraic Effects" 2009

    Bauer, Pretnar "Progamming with algebraic effects and handlers" 2012]

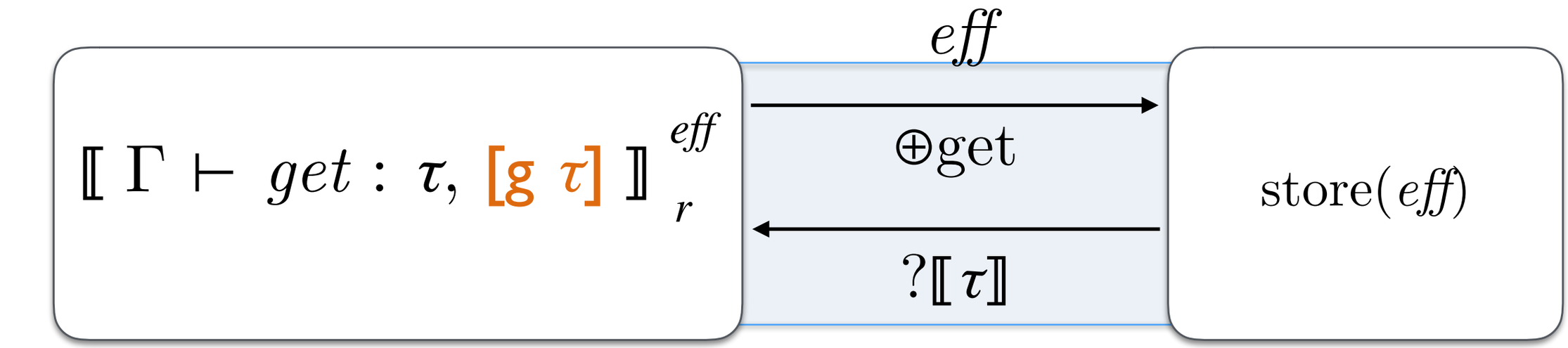
  - ➡ **Interpretation of effects annotations into sessions**

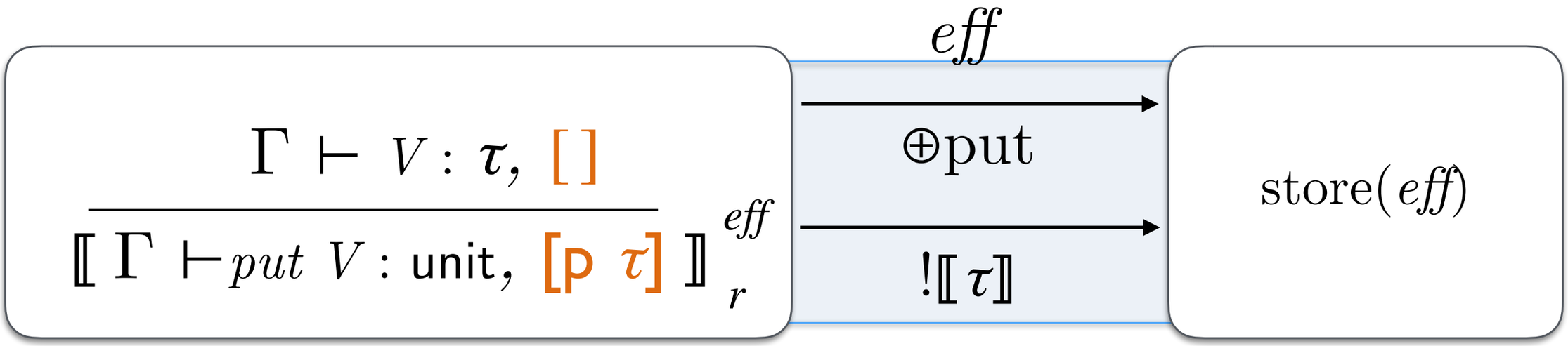
    $[\![\ \_\ ]\!] : \mathsf{F} \to S$ satisfying

    $$[\![\ \mathsf{F} \bullet \mathsf{G}\ ]\!] = [\![\ \mathsf{F}\ ]\!] \mathbin{\blacklozenge} [\![\ \mathsf{G}\ ]\!]$$
    $$[\![\ \mathsf{I}\ ]\!] = \mathbf{end}$$

  - ➡ **Encoding of effectful operations**
    - **(e.g.,** get, put, input, ouput**)**

# Simple state effects (List {p $t$, g $t$ | $t \in \tau$}, ++ , [])




⟦ _ ⟧ : F → $S$ for simple state

$$⟦ (g\ \tau) :: F ⟧ = ⊕[\text{get} : ?⟦\tau⟧.\ ⟦ F ⟧ ]$$
$$⟦ (p\ \tau) :: F ⟧ = ⊕[\text{put} : !⟦\tau⟧.\ ⟦ F ⟧ ]$$
$$⟦ [] ⟧ = \mathbf{end}$$

$$\text{let}\ \frac{\Gamma \vdash M : \sigma,\ \mathsf{F} \qquad \Gamma,\ x : \sigma \vdash N : \tau,\ \mathsf{G}}{\Gamma \vdash \mathbf{let}\ x \Leftarrow M\ \mathbf{in}\ N : \tau,\ (\mathsf{F} \bullet \mathsf{G})}$$

**syntactic sugar for** $(\lambda x.M)\ N$

$$[\![\ \Gamma \vdash M : \sigma,\ \mathsf{F}\ ]\!]^{eff}_{r} \longrightarrow \ldots.\ eff : [\![\ \mathsf{F}\ ]\!] \vdash P$$

$$[\![\ \Gamma,\ x : \sigma \vdash N : \tau,\ \mathsf{G}\ ]\!]^{eff}_{r} \longrightarrow \ldots.\ eff : [\![\ \mathsf{G}\ ]\!] \vdash P'$$

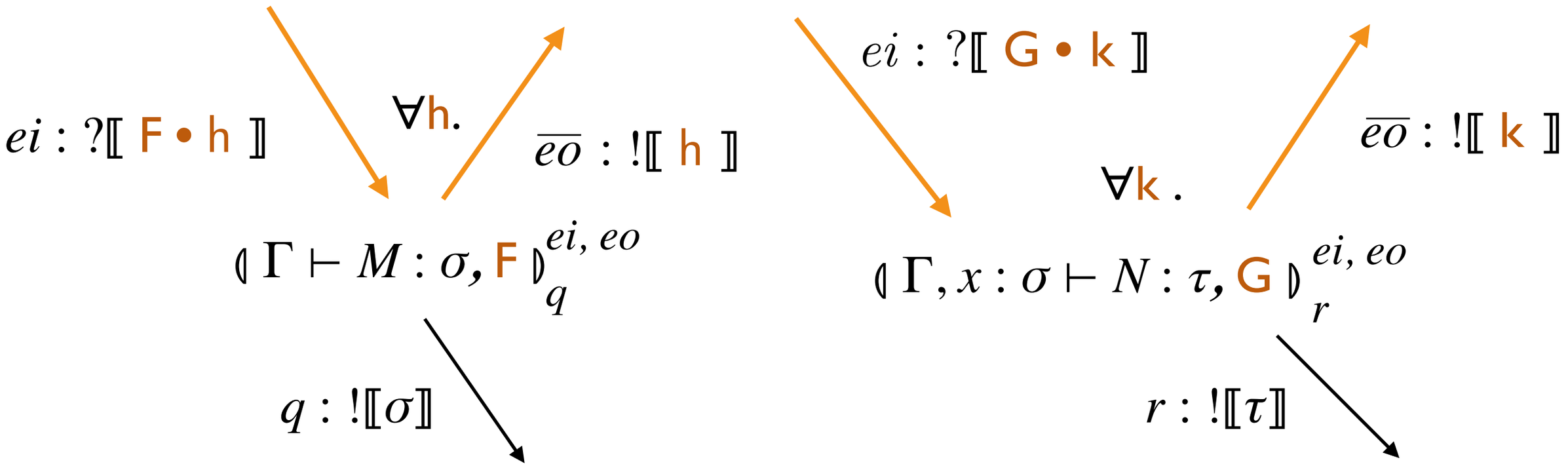
---

$$[\![\ \Gamma \vdash \mathbf{let}\ \ldots : \tau,\ (\mathsf{F} \bullet \mathsf{G})\ ]\!]^{eff}_{r} \longrightarrow \ldots.\ eff : [\![\ \mathsf{F} \bullet \mathsf{G}\ ]\!] \vdash \mathbf{???}$$

Solution: pass *eff around instead*

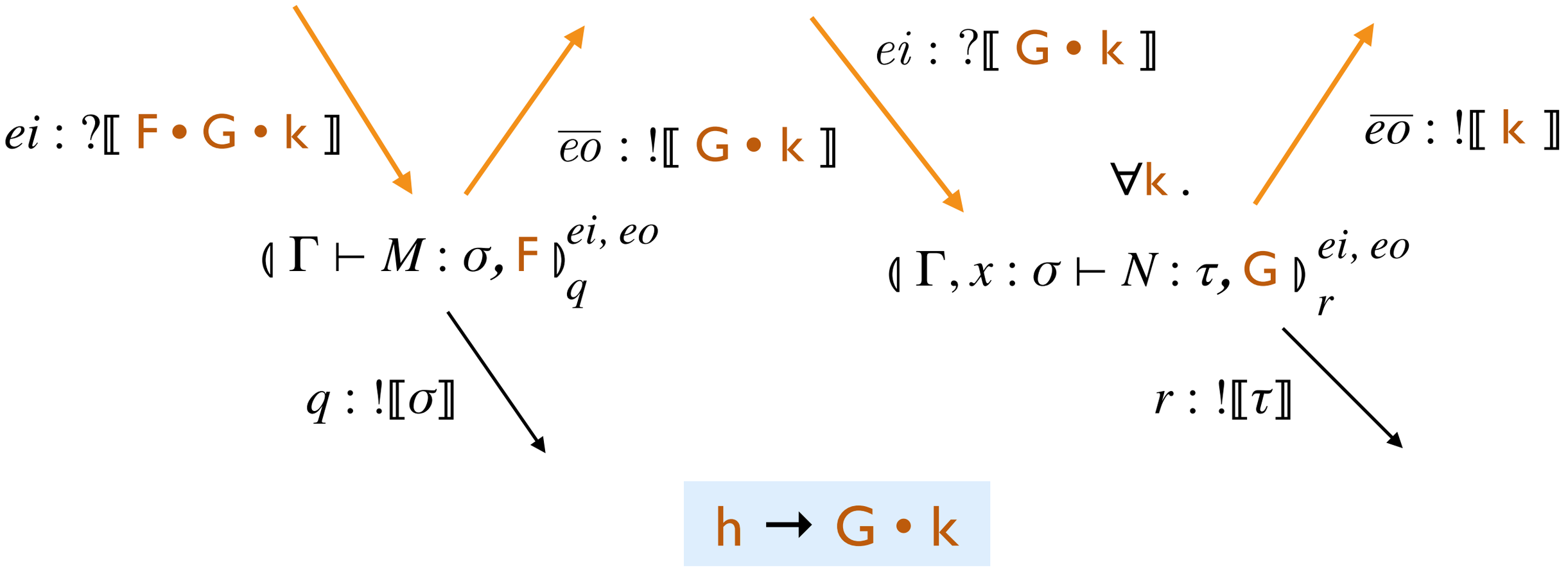
# Effect-channel passing

$$\text{let}\ \frac{\Gamma \vdash M : \sigma,\ \mathsf{F} \qquad \Gamma,\ x : \sigma \vdash N : \tau,\ \mathsf{G}}{\Gamma \vdash \mathbf{let}\ x \Leftarrow M\ \mathbf{in}\ N : \tau,\ (\mathsf{F} \bullet \mathsf{G})}$$



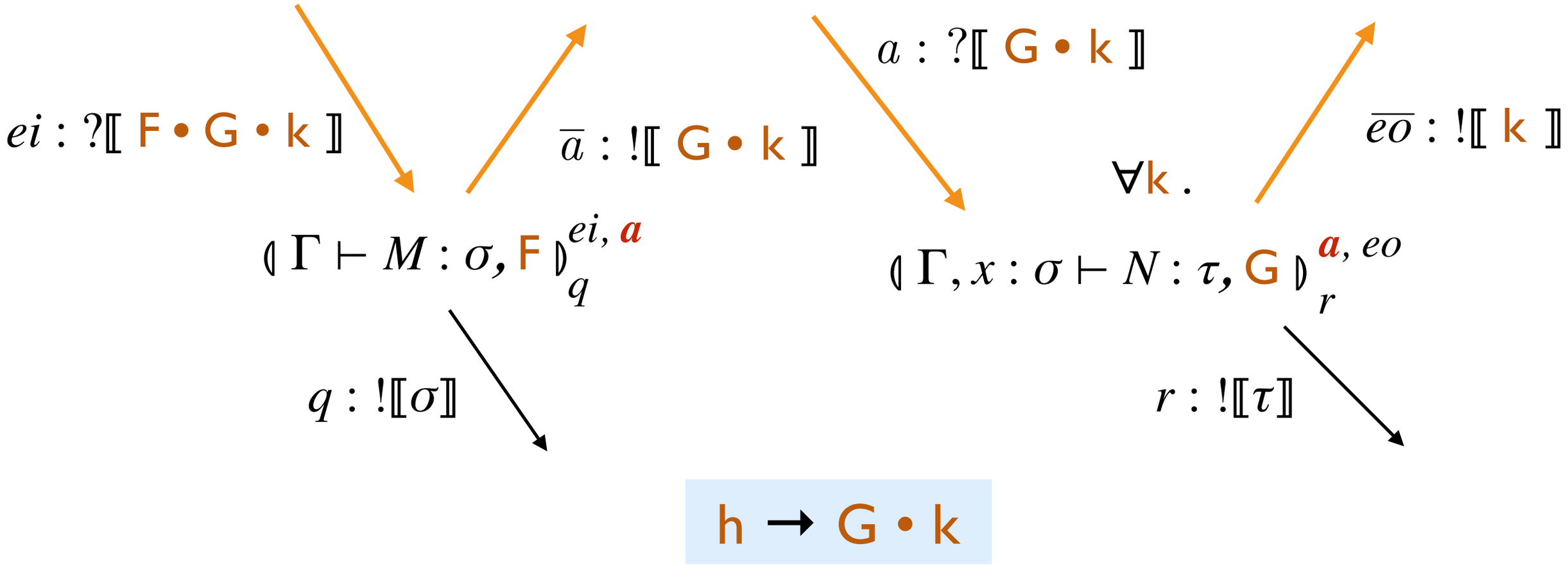
# Effect-channel passing

$$\text{let}\ \frac{\Gamma \vdash M : \sigma,\ \mathsf{F} \qquad \Gamma,\ x : \sigma \vdash N : \tau,\ \mathsf{G}}{\Gamma \vdash \mathbf{let}\ x \Leftarrow M\ \mathbf{in}\ N : \tau,\ (\mathsf{F} \bullet \mathsf{G})}$$



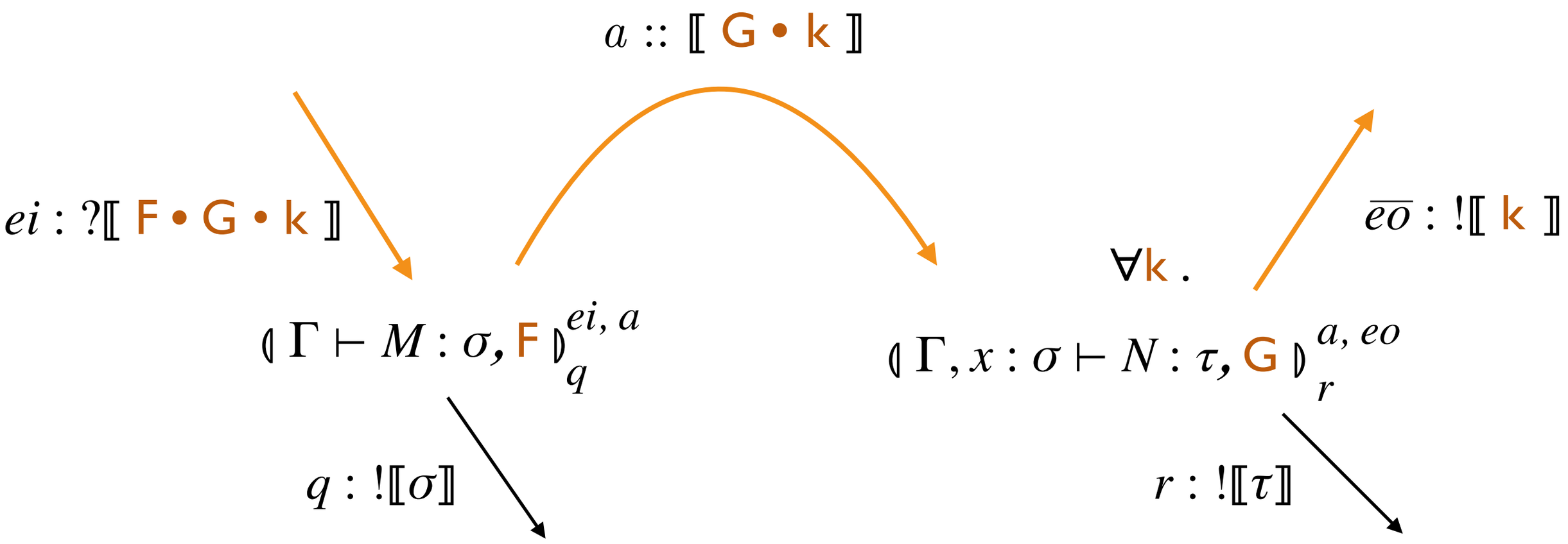
# Effect-channel passing

$$\text{let}\ \frac{\Gamma \vdash M : \sigma,\ \mathsf{F} \qquad \Gamma,\ x : \sigma \vdash N : \tau,\ \mathsf{G}}{\Gamma \vdash \mathbf{let}\ x \Leftarrow M\ \mathbf{in}\ N : \tau,\ (\mathsf{F} \bullet \mathsf{G})}$$



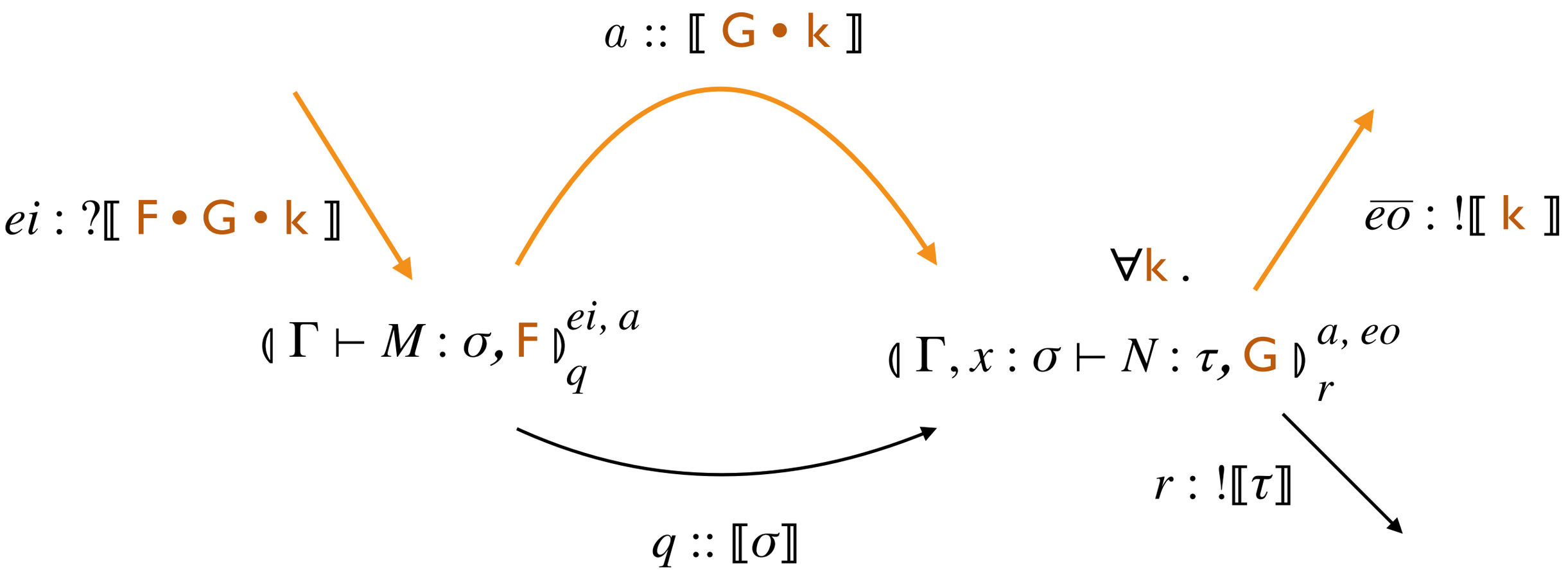
# Effect-channel passing

$$\text{let}\ \frac{\Gamma \vdash M : \sigma,\ \mathsf{F} \qquad \Gamma,\, x : \sigma \vdash N : \tau,\ \mathsf{G}}{\Gamma \vdash \mathbf{let}\ x \Leftarrow M\ \mathbf{in}\ N : \tau,\ (\mathsf{F} \bullet \mathsf{G})}$$



# Effect-channel passing

$$\text{let}\ \frac{\Gamma \vdash M : \sigma,\ \mathsf{F} \qquad \Gamma,\ x : \sigma \vdash N : \tau,\ \mathsf{G}}{\Gamma \vdash \mathbf{let}\ x \Leftarrow M\ \mathbf{in}\ N : \tau,\ (\mathsf{F} \bullet \mathsf{G})}$$




$$r : ![\![\tau]\!],\ ei : ?[\![ \mathsf{F} \bullet \mathsf{G} \bullet \mathsf{k} ]\!],\ \overline{eo} : ![\![ \mathsf{k} ]\!] \vdash (\!| \Gamma \vdash \mathbf{let}\ \ldots. : \tau, (\mathsf{F} \bullet \mathsf{G}) |\!)^{ei,\ eo}_{r}$$

# Higher-order embedding

Embed *latent effects* $\sigma \xrightarrow{\mathsf{F}} \tau$

$$[\![\ \sigma \to \tau\ ]\!] = \quad !\big[?[\![\ \sigma\ ]\!],\ ![\![\tau]\!]\ \big]$$

$$[\![\ \sigma \xrightarrow{\mathsf{F}} \tau\ ]\!] = \quad !\big[?[\![\ \sigma\ ]\!],\ ?[\![\ \mathsf{F} \bullet \mathsf{G}\ ]\!],\ ![\![\ \mathsf{G}\ ]\!],\ \ ![\![\tau]\!]\ \big]$$

send channel which
can receive effect channel
for latent effects

send channel which
can send effect channel
for **continuation**

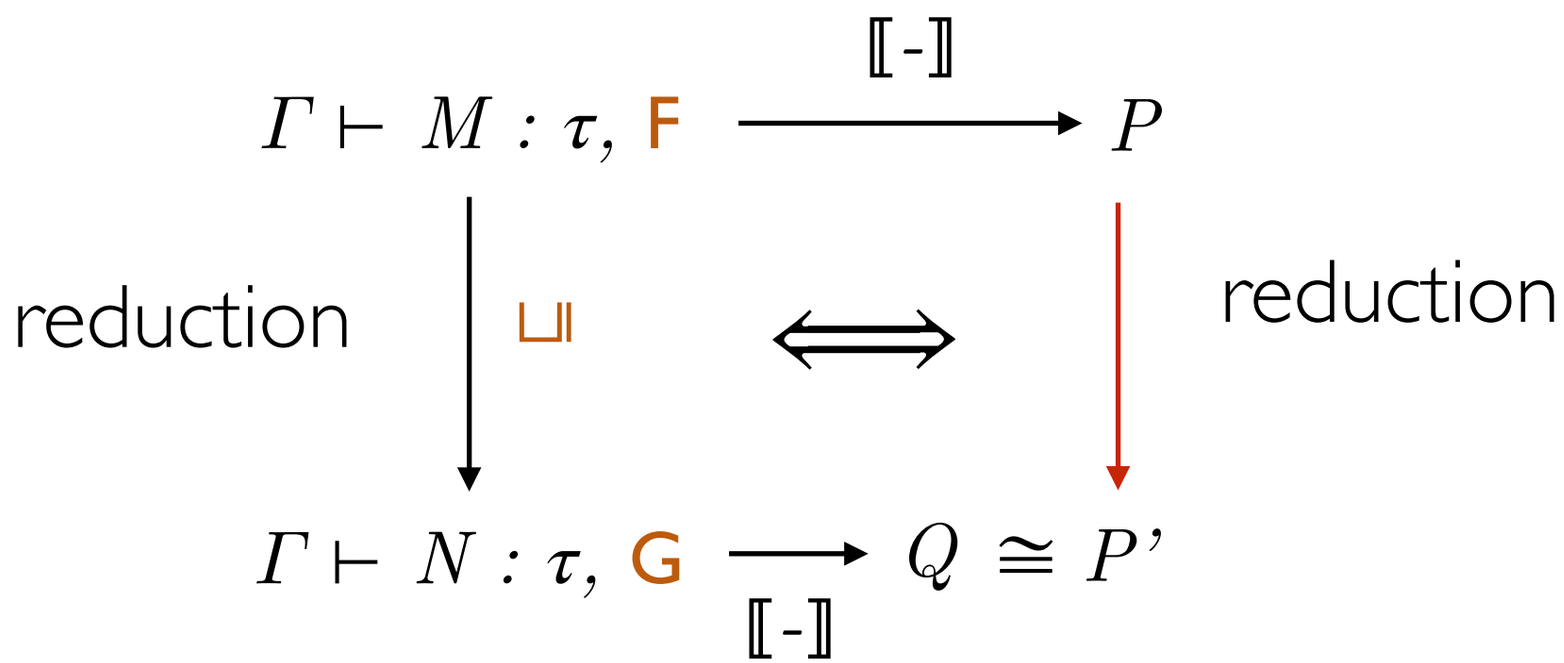
# Theorems

## Soundness

$$\Gamma \vdash M = N : \tau, \mathsf{F} \implies$$

$$[\![ \Gamma ]\!], r : ![\![\tau]\!], \mathit{eff} : [\![ \mathrm{F} ]\!] \vdash [\![ M ]\!] \cong [\![ N ]\!]$$

## Operational correspondence




Soundness/completeness

# Effects as sessions

$$[\![\ \Gamma \vdash M : \tau,\ \mathsf{F}\ ]\!]^{\mathit{eff}}_{r} \longrightarrow [\![\ \Gamma\ ]\!], r : ![\![\tau]\!],\ \mathit{eff} : [\![\ \mathsf{F}\ ]\!] \vdash P$$

- Encoding parameterised by:
  - handler, (homomorphic) effect interp., operations encoding
- Examples
  - State (list and set-based)
  - ML-style references
  - Counting effects, $\mathsf{F} = \mathbb{N}$
- Effects with linear continuation behaviour

# Sessions as effects

$$[\![\ \Gamma\ ]\!] \vdash M : \textbf{unit}, [\![\ \Delta\ ]\!] \longleftarrow \Gamma; \Delta \vdash P$$

# Effects vs. session types

| effect systems | | session types | |
|---|---|---|---|
| $\bullet$ | sequential | $?[\tau].\text{-}$ $\quad$ $![\tau].\text{-}$ | prefixing |
| $\mathrm{I}$ | pure | $\mathbf{end}$ | inaction |
| $\oplus$ | conditional | $\oplus[\mathrm{l}_1 : S_1, \ \ldots \ \mathrm{l}_\mathrm{n} : S_\mathrm{n}]$ | selection/ |
| | | $\&[\mathrm{l}_1 : S_1, \ \ldots \ \mathrm{l}_\mathrm{n} : S_\mathrm{n}]$ | branching |
| $*$ | repetition | $\mu a \ . \ S$ $\quad$ $a$ | fixed-points |
| | | $*![\tau]$ $\ *?[\tau]$ | replication |
| $\sqsubseteq$ | sub-effecting | $\leqq$ | sub-typing |
| $\&$ | parallelism | $\odot$ | balancing |

RECAP

# Sessions into effects

1. PCF with par

$$\text{par}\ \frac{\Gamma \vdash M : \text{unit},\ \mathsf{F} \qquad \Gamma \vdash N : \textbf{unit},\ \mathsf{G}}{\Gamma \vdash M \mid N : \text{unit},\ \mathsf{F}\ \&\ \mathsf{G}}$$

2. Instantiate PCF effects with 'session effects'
   - F = FiniteMap Chan S (effects are environments **Δ**)
   - algebra replicates session operators, e.g., $\& = \odot$
3. **Channel types** ($Chan\ c$)
4. **π-calculus like communication operations** (send, recv, new)
5. **Operational semantics based on queues:**

$$<M,\ s>\quad \textit{where s is a store for queues}$$

# Sessions into effects

Sequential composition (fragment)

$$(\{c : \mathsf{S}\},\ \Delta)\ \bullet\ (\{c : \mathsf{T}\}\ ,\ \Delta')\ =\ \{c : \mathsf{S} \mathbin{✦} \mathsf{T}\}\ ,\ (\Delta \bullet \Delta')$$

Communication operations

$$recv : Chan\ c \xrightarrow{\{c\ :\ ?[\tau].\mathbf{end}\}} \tau$$

$$send : Chan\ c \to\ \tau \xrightarrow{\{c\ :\ ![\tau].\mathbf{end}\}} \mathsf{unit}$$

$$chSend : Chan\ c \to\ Chan\ d \to\ \tau \xrightarrow{\{c\ :\ ![S].\mathbf{end},\ d\colon S\}} \mathsf{unit}$$

$$chRecv : Chan\ c \xrightarrow{\{c\ :\ ?[S]\}} (Chan\ d \xrightarrow{\mathsf{F} \bullet \{d\ :\ S\}} \tau\ ) \xrightarrow{\mathsf{F}} \tau$$

$$new : (Chan\ c \to\ Chan\ \overline{c} \xrightarrow{\mathsf{F} \bullet \{c\ :\ S,\ \overline{c}\ :\ \mathrm{dual}(S)\}} \tau\ ) \xrightarrow{\mathsf{F}} \tau$$

# Sessions into effects

Encoding of session types into effects: $[\![ \_ ]\!] : S \to \mathsf{F}$

**e.g.**

$$[\![ ![\tau].S ]\!] = ![\![\tau]\!] \,.\, [\![S]\!]$$
$$[\![ ?[\tau].S ]\!] = ?[\![\tau]\!] \,.\, [\![S]\!]$$
$$[\![ \&[\mathsf{l}_1 : S_1, \mathsf{l}_2 : S_2] ]\!] = ?[\mathsf{int}] \,.\, ([\![S_1]\!] + [\![S_2]\!])$$
$$[\![ \oplus[\mathsf{l}_1 : S_1, \mathsf{l}_2 : S_2] ]\!] = ![\mathsf{int}] \,.\, ([\![S_1]\!] + [\![S_2]\!])$$

Choice/selection via alternation

$$\mathsf{S}, \mathsf{T} ::= ![\tau].\mathsf{S} \mid ?[\tau].\mathsf{S} \mid *![\tau].\mathsf{S} \mid \mathbf{end} \mid \mathsf{S} + \mathsf{S} \mid \mu\alpha.\mathsf{S} \mid \alpha \mid \odot\mathsf{S}$$

Effects of case $\oplus$ = union if common session types equal

$$\Gamma \vdash \mathbf{case}\ M\ \mathbf{of}\ 0 \mapsto N_1, (\mathsf{suc}\ x) \mapsto N_2 : \tau,\ \mathsf{F} \bullet (\mathsf{G} \cup \mathsf{H})$$

Sub-typing used to introduce branch/select

$$\mathsf{S} \sqsubseteq \mathsf{S} + \mathsf{T} \qquad \mathsf{T} \sqsubseteq \mathsf{S} + \mathsf{T}$$

# Sessions into effects (syntax)

Fragment of the encoding…. mostly straightforward

$$[\![\ c!\langle V\rangle.P\ ]\!] = \mathbf{let}\ \_ = send\ c\ [\![\ V\ ]\!]\ \mathbf{in}\ [\![\ P\ ]\!]$$

$$[\![\ c?(x).P\ ]\!] = \mathbf{let}\ x = recv\ c\ \mathbf{in}\ [\![\ P\ ]\!]$$

$$[\![\ c?(d).P\ ]\!] = \mathbf{let}\ k = chRecv\ c\ \mathbf{in}\ k\ (\lambda d.\ [\![\ P\ ]\!])$$

$$chRecv : Chan\ c \xrightarrow{\{c\,:\,?[S]\}} (Chan\ d \xrightarrow{\mathsf{F}\,\bullet\,\{d\,:\,S\}} \tau) \xrightarrow{\mathsf{F}} \tau$$

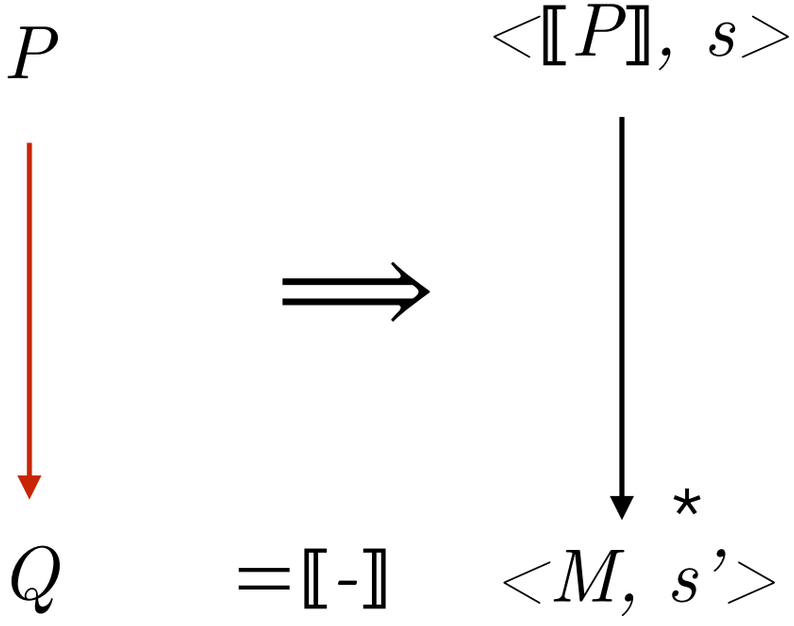
$$[\![\ c \triangleleft \mathrm{l}_0.P\ ]\!] = \mathbf{let}\ \_ = send\ c\ 0\ \mathbf{in}\ [\![\ P\ ]\!]$$

$$[\![\ c \triangleright [\mathrm{l}_0 : P, \mathrm{l}_1 : Q]\ ]\!] = \mathbf{let}\ x = recv\ c\ \mathbf{in}\ \mathbf{case}\ x\ \mathbf{of}\ 0 \mapsto [\![\ P\ ]\!],\ (\mathbf{suc}\ n) \mapsto [\![\ Q\ ]\!]$$

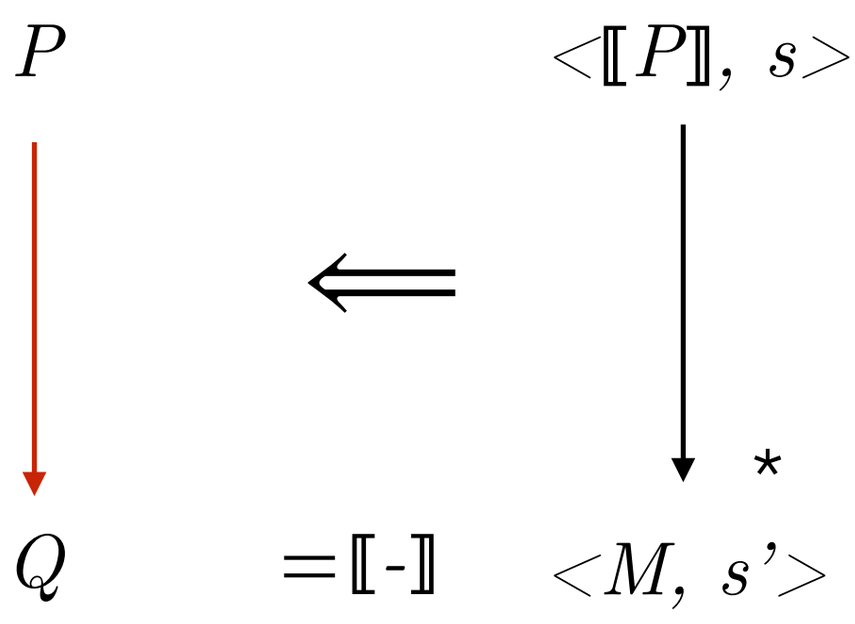
$$[\![\ *c?(d).P\ ]\!] = \mathbf{rec}\ (\lambda f.\lambda x.\ \mathbf{let}\ k = chRecv\ c\ \mathbf{in}\ (k\ (\lambda d.\ [\![\ P\ ]\!]))\ ||\ f\ \mathsf{unit})\ \mathsf{unit}$$

# Theorems

## Operational correspondence




Soundness




Completeness

# Prototype implementation


Sessions $\rightarrow$ Effect PCF

\+ “Embedding effect systems in Haskell”, Orchard, Petricek (2014)

Operations

```
recv :: Chan c → Process '[c ↦ t :? end] t
send :: Chan c → t → Process '[c ↦ t :! end] ()
```

Effect-graded monad

```
return :: a → Process (Unit t) a
(>>=) :: t f a → (a → t g b)
                → Process (Plus t f g) b
```

http://dorchard.co.uk/popl16

# Encoding of: $\nu c.(\nu d.(c!\langle d\rangle.\overline{d}?(\mathsf{Ping})) \mid \overline{c}?(x).x!\langle\mathsf{Ping}\rangle)$

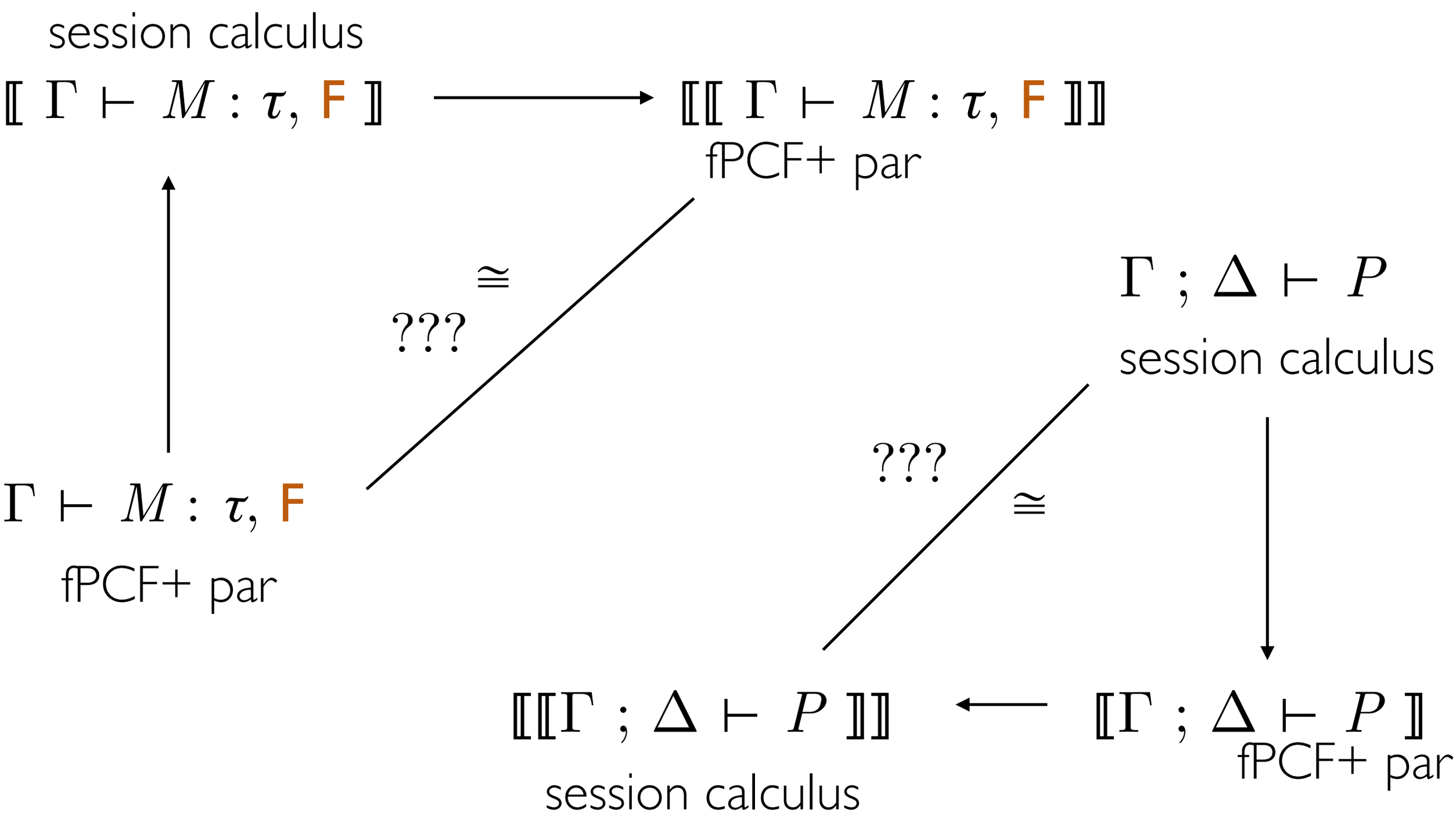
```
client (c :: (Chan (Ch "c")))
    = new \(d :: (Chan (Ch "d")), d') ->
                  do  chSend c d
                      Ping <- recv d'
                      print "Client: got a ping")
```

```
server c = do { k <- chRecv c; k (\x -> send x Ping) }
process = new (\(c, c') -> (client c) `par` (server c'))
```

Inferred type:

```
client :: Chan (Ch "c")
-> Process '[Ch "c" :-> (Delg (Msg :! End) :! End)] ()
```

# Unanswered question



# Conclusion

Effects (parallel PCF) into sessions ($\pi$-calculus)

- **Expressive power of session types**
- **Incorporate effect information into specifications (e.g. Scribble)**
- **Pi-calculus as intermediate language**

Sessions ($\pi$-calculus) into effects (parallel PCF)

- Expressive power of effect typing
- Embed session types into existing languages (see artefact)

Todo: Completness

Alternate encoding: $\Gamma \vdash M : \mathrm{T}_{\mathsf{F}}\, \tau$ *or* *CBPV*

# Thanks

http://dorchard.co.uk/popl16