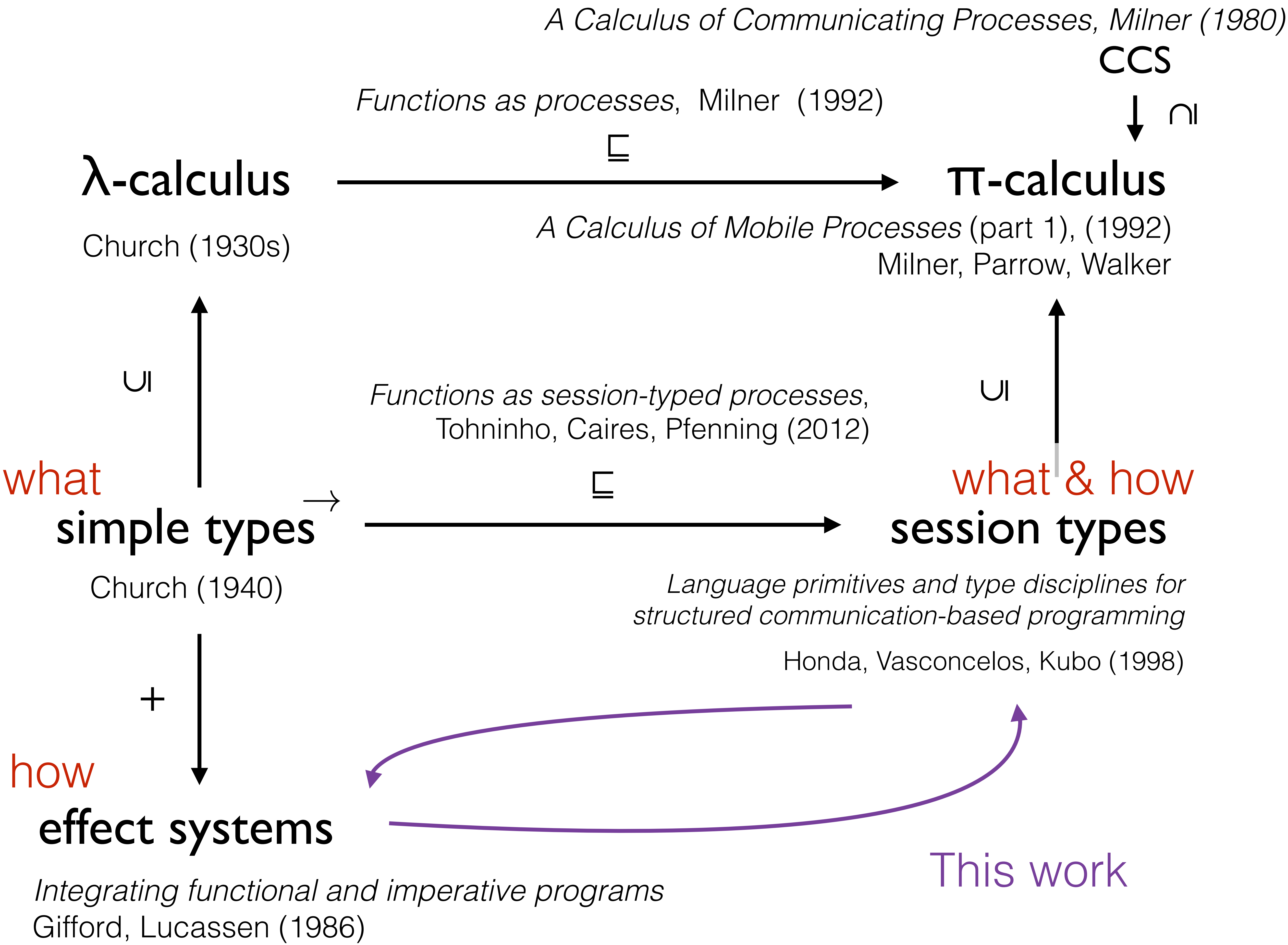
# Sessions as effects; effects as sessions

## the tale of two type systems


Dominic Orchard

Imperial College London

dorchard.co.uk


A Calculus of Communicating Processes, Milner (1980)
CCS
Functions as processes, Milner (1992)
λ-calculus
π-calculus
A Calculus of Mobile Processes (part 1), (1992)
Milner, Parrow, Walker
Church (1930s)
Functions as session-typed processes,
Tohninho, Caires, Pfenning (2012)
what
what & how
simple types
session types
Church (1940)
Language primitives and type disciplines for
structured communication-based programming
Honda, Vasconcelos, Kubo (1998)
+
how
effect systems
This work
Integrating functional and imperative programs
Gifford, Lucassen (1986)

# π-calculus primer

| | |
|---|---|
| $c!\langle V\rangle.P$ | send $V$ on $c$ then act as $P$ |
| $c?(x).P$ | receive on $c$, bind to $x$, then act as $P$ |
| $P \mid Q$ | do $P$ and $Q$ in parallel |
| $\nu c\ (P)$ | channel creation/binding [restriction] |
| $!\ P$ | process replication |
| $\mathbf{0}$ | inactive process |

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

(commutativity)

$$P \mid Q = Q \mid P$$

(associativity)

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

(scope extrusion)

$$\nu c\ (P \mid Q) = \nu c\ (P) \mid Q \quad (\textit{if}\ c\#Q)$$

# Session primitives primer

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

$c \lhd L_i \,.\, P$ — select label *i* then act as P

**dual end-point**

$$(c \rhd \{L_1 : P_1, \ldots, L_n : P_n\} \;|\; \underline{c} \lhd L_i \,.\, Q) \quad \rightarrow \quad (P_i \;|\; Q)$$

(β reduction)

$$(c?(x).P \;|\; \underline{c}!\langle V \rangle.Q) \quad \rightarrow \quad (P[V/x] \;|\; Q)$$

(β reduction)

**dual end-point**

# Session types primer

$$\Gamma\ ;\ \Delta \vdash \mathrm{P}$$

**value environment** $x_1 : A_1,\ \ldots,\ x_n : A_n$

**session environment** $c_1 : S_1,\ \ldots,\ c_n : S_n$

$$\text{(recv)}\ \frac{\Gamma, x : A\ ;\ \Delta, c : S \vdash P}{\Gamma\ ;\ \Delta, c : ?[A].S \vdash c?(x).P}$$

$$\text{(send)}\ \frac{\Gamma\ ;\ \Delta, c : S \vdash P \qquad \Gamma; . \vdash V : A}{\Gamma\ ;\ \Delta, c : ![A].S \vdash c!\langle V\rangle.P}$$

# Session types primer (2)

dual session type

(inact) $\Gamma;\ c : \mathbf{end} \vdash \mathbf{0}$

(restr) $$\frac{\Gamma\,;\Delta,\ c : S,\ \underline{c} : \underline{S} \vdash\ P}{\Gamma\ ;\ \Delta \vdash \nu c.P}$$

**Duality:** *ensures absence of communication errors*

$\underline{?[A].S} = ![A].\underline{S}$ $\quad \underline{![A].S} = ?[A].\underline{S}$ $\quad \underline{\mathbf{end}} = \mathbf{end}$

e.g. ~~$c!\langle 0\rangle.c!\langle 1\rangle \mid \underline{c}?(x)$~~ $\quad$ ~~$c!\langle 0\rangle \mid \underline{c}?(x).\underline{c}?(y) \mid c!\langle 1\rangle$~~

(par) $$\frac{\Gamma\,;\ \Delta_1 \vdash\ P \qquad \Gamma\,;\ \Delta_2 \vdash\ Q}{\Gamma\,;\ \Delta_1 \odot \Delta_2 \vdash\ P \mid Q}$$

# Session types primer (2)

dual session type →

(inact) $\Gamma;\ c : \mathbf{end} \vdash \mathbf{0}$

(restr) $$\frac{\Gamma\,;\Delta,\ c : S,\ \underline{c} : \underline{S} \vdash\ P}{\Gamma\ ;\ \Delta \vdash \nu c.P}$$

**Duality:** *ensures absence of communication errors*

$$\underline{?[A].S} = ![A].\underline{S} \qquad \underline{![A].S} = ?[A].\underline{S} \qquad \underline{\mathbf{end}} = \mathbf{end}$$

e.g. ~~$c!\langle 0\rangle.c!\langle 1\rangle \mid \underline{c}?(x)$~~ $\qquad c!\langle 0\rangle \mid \underline{c}?(x).\underline{d}?(y) \mid d!\langle 1\rangle$ ✔

(par) $$\frac{\Gamma\,;\ \Delta_1 \vdash\ P \qquad \Gamma\,;\ \Delta_2 \vdash\ Q}{\Gamma\,;\ \Delta_1 \odot \Delta_2 \vdash\ P \mid Q}$$

# Session types primer (3)

(branch)
$$\frac{\Gamma;\Delta,\ c:S_0 \vdash P_0 \quad \ldots \quad \Gamma;\Delta,\ c:S_n \vdash P_n}{\Gamma\ ;\ \Delta,\ c:\&[l_0:S_0 \ldots\ l_n:S_n] \vdash c \triangleright \{l_0:P_0 \ldots\ l_n:P_n\}}$$

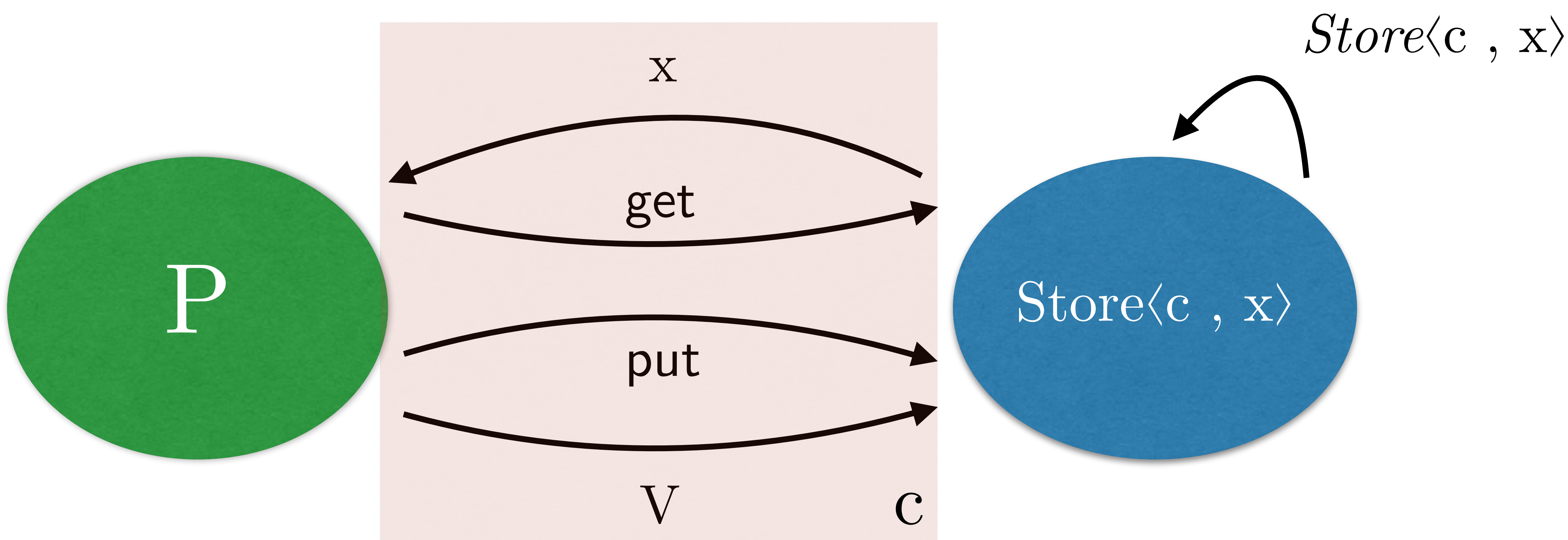
(select)
$$\frac{\Gamma\ ;\ \Delta,\ c:S \vdash\ P}{\Gamma\ ;\ \Delta,\ c:\oplus[l:S] \vdash c \triangleleft\ l\ .P}$$

Duality:

$$\underline{\&[l_0:S_0 \ldots\ l_n:S_n]} = \oplus[l_0:\underline{S_0} \ldots\ l_n:\underline{S_n}]$$

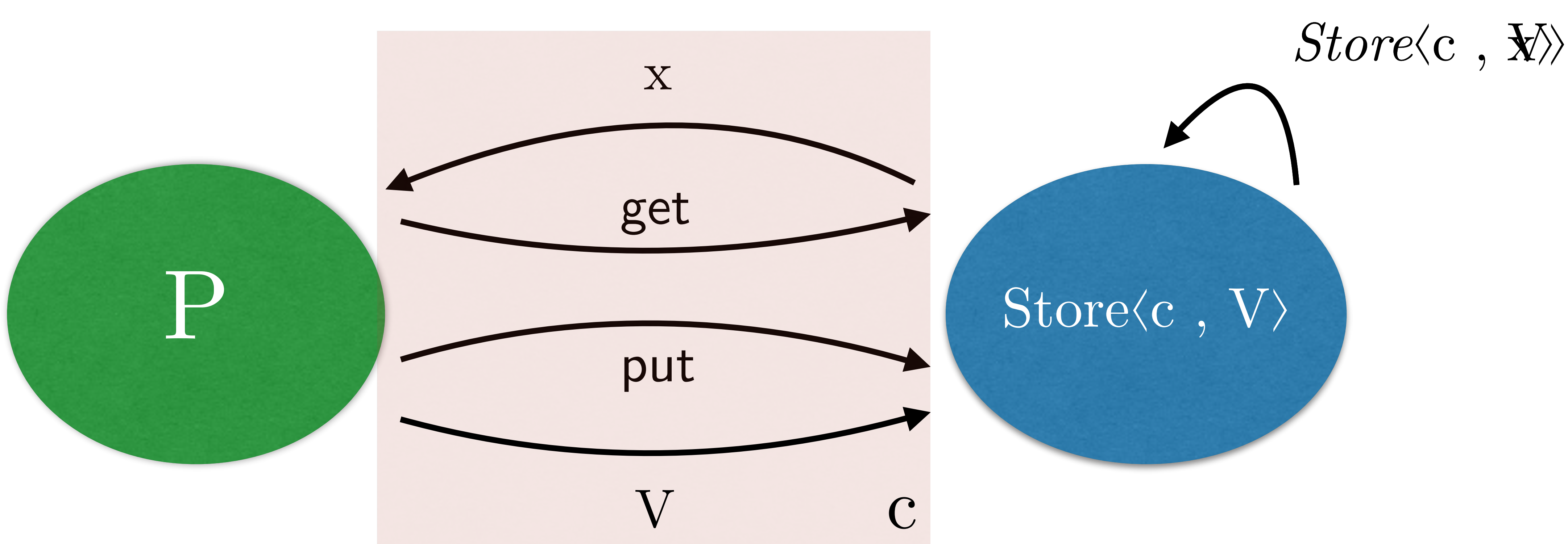
$$\underline{\oplus[l_0:S_0 \ldots\ l_n:S_n]} = \&[l_0:\underline{S_0} \ldots\ l_n:\underline{S_n}]$$

# Effects in a π : *Variable agent*



# Effects in a π : *Variable agent*

$$\textbf{def } \mathit{Store}(c, x) = c \triangleright \{\mathsf{get} : c!\langle x\rangle.\mathit{Store}\langle c, x\rangle,\ \mathsf{put} : c?(y).\mathit{Store}\langle c, y\rangle,\ \mathsf{stop} : \mathbf{0}\} \quad \textbf{in } \mathit{Store}\langle c\ ,\ i\rangle$$



# Effects in a π : *Variable agent*

Server

$$\textbf{def}\ \mathit{Store}(c, x) = c \rhd \{\text{get} : c!\langle x\rangle.\mathit{Store}\langle c, x\rangle,$$
$$\text{put} : c?(y).\mathit{Store}\langle c, y\rangle,$$
$$\text{stop} : \mathbf{0}\} \quad \textbf{in}\ \mathit{Store}\langle c\ ,\ i\rangle$$

Client

$$\text{get}(c)(x).P = (c \lhd \text{get}).c?(x).P$$
$$\text{put}(c)\langle V\rangle.P = (c \lhd \text{put}).c!\langle V\rangle.P$$
$$\text{stop} = (c \lhd \text{stop}).\mathbf{0}$$

$c : \oplus[\text{get} : ?[A].\ S]$

$c : \oplus[\text{put} : ![A].\ S]$

**session types**

e.g. increment store

$$(\text{get}(c)(x).\text{put}(c)\langle x+1\rangle.\mathbf{0} \mid \mathit{Store}\langle \underline{c}\ ,\ i\rangle)$$

$$c : \oplus[\text{get} : ?[\mathbb{Z}].\ \oplus[\text{put} : ![\mathbb{Z}].\textbf{end}\ ]] \ \vdash\ \text{get}(c)(x).\text{put}(c)\langle x+1\rangle.\mathbf{0}$$

**describes effect interaction**

$\Gamma \vdash \mathrm{M} : \tau$

# Effect system

monoid $(F, \bullet, \varnothing)$

abs $$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x.M : \sigma \to \tau}$$

app $$\frac{\Gamma \vdash M : \sigma \to \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash M\ N : \tau}$$

var $$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}$$

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

return $$\frac{\Gamma \vdash M : \tau}{\Gamma \vdash <M> : \mathsf{T}\ \varnothing\ \tau}$$

*The marriage of effects and monads,* Wadler & Thieman (1992)

***Analysis*** **style** [Gifford, Lucassen (1986), etc.]

$$\frac{\Gamma, x : \sigma \vdash M : \tau, F}{\Gamma \vdash \lambda x.M : \sigma \xrightarrow{F} \tau, \varnothing}$$

$$\frac{\Gamma \vdash M : \sigma \xrightarrow{H} \tau, F \qquad \Gamma \vdash N : \sigma, G}{\Gamma \vdash M\ N : \tau, F \bullet G \bullet H}$$

# Effect system for state

Effect monoid: $(\text{List } \{\mathsf{put}\ \boldsymbol{t}, \mathsf{get}\ \boldsymbol{t} \mid \boldsymbol{t} \in \tau\}, ++, [\,])$

Effect operations:

$\mathsf{put} : \tau \to \mathsf{T}\ [\mathsf{put}\ \tau]\ ()$

$\mathsf{get} : \mathsf{T}\ [\mathsf{get}\ \tau]\ \tau$

e.g. increment store

$$\Gamma \vdash \mathbf{let}\ x \Leftarrow \mathsf{get}\ \mathbf{in}\ \mathsf{put}\ (x+1) : \mathsf{T}\ [\mathsf{get}\ \mathbb{Z},\ \mathsf{put}\ \mathbb{Z}]\ ()$$

cf. session-typed π-calculus version

$$\mathrm{c} : \oplus[\mathsf{get} : ?[\mathbb{Z}].\ \oplus[\mathsf{put} : ![\mathbb{Z}].\mathbf{end}\ ]] \vdash \mathsf{get}(\mathrm{c})(\mathrm{x}).\mathsf{put}(\mathrm{c})\langle \mathrm{x}+1\rangle.\mathbf{0}$$

$$\simeq \quad [\mathsf{get}\ \mathbb{Z},\ \mathsf{put}\ \mathbb{Z}]$$

# Effect systems describe side-effect *behaviour*

λ-calculus as prototype

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

# Session types describe communication *behaviour*

π-calculus as prototype

$$(\mathrm{recv})\quad \frac{\Gamma, x : \tau; \Delta, c : S \vdash P}{\Gamma; \Delta, c : ?[\tau].S \vdash c?(\mathrm{x}).\mathrm{P}}$$

# Are they related?

# Sessions as effects

- ***Effect handler*** process [e.g., variable agent]

  [cf. Bauer, Pretnar "Progamming with algebraic effects and handlers."]

- ***Effect channel*** [a session channel for communicating with handler]

  … whose session type is (encoding of) effect annotation

- "Threading" effect channel through control flow of encoding

  [cf. state $\langle e, s\rangle \rightarrow \langle e', s'\rangle$ or monadic semantics $a \rightarrow M\ b$ ]

# Sessions as effects

- Encoding of general effect sequential composition
- Parameterised, for a particular notion of effect by:
  - *Effect handler* [variable agent]
  - Interpretation of effects annotations into sessions

$$[\![\ [] \ ]\!] = \mathbf{end}$$

$$[\![\ (\mathrm{get}\ \tau) : F\ ]\!] = \oplus[\mathrm{get} : ?[\![\tau]\!].\ [\![\ F\ ]\!]\ ]$$

$$[\![\ (\mathrm{put}\ \tau) : F\ ]\!] = \oplus[\mathrm{put} : ![\![\tau]\!].\ [\![\ F\ ]\!]\ ]$$

  - Encoding of operations [get, put]
    - using get(c)(x).P = (c ◁ get).c?(x).P etc.

# Embedding

$$[\![\ \Gamma \vdash \mathrm{M} : \mathsf{T}\ \mathsf{F}\ \tau\ ]\!]_r^{\mathit{eff}} = [\![\ \Gamma\ ]\!], r : ![\![\tau]\!],\ \mathit{eff} : [\![\ \mathsf{F}\ ]\!] \vdash\ \ldots$$

$$[\![\ \Gamma \vdash \mathbf{let}\ x \Leftarrow M\ \mathbf{in}\ N\ : \mathsf{T}\ \mathsf{F}\ \tau\ ]\!]_r^{\mathit{eff}} =$$

$$\nu s\ .\ (\ [\![\ M\ ]\!]_s \mid \underline{s}?(x).[\![\ N\ ]\!]_r\ )$$

both using $\mathit{eff}$

**Not well-typed wrt. sessions!**

# Embedding

$$[\![ \Gamma \vdash \mathrm{M} : \mathbf{T}\ \mathsf{F}\ \tau ]\!]_r^{eff} = [\![ \Gamma ]\!], r : ![\![\tau]\!], eff : [\![ \mathsf{F} ]\!] \vdash \ldots$$

$$\nu ei, eo\ .\ ((\!| \Gamma \vdash \mathrm{M} : \mathbf{T}\ \mathsf{F}\ \tau |\!)_r^{ei,eo} \mid \underline{ei}!\langle eff\rangle.eo(eff'))$$

$$(\!| \Gamma \vdash \mathrm{M} : \mathbf{T}\ \mathsf{F}\ \tau |\!)_r^{ei,eo} = \forall \mathsf{g}\ .$$

$$[\![ \Gamma ]\!], r : ![\![\tau]\!], ei : ?[\![ \mathsf{F} \bullet \mathsf{g} ]\!], \underline{eo} : ![\![ \mathsf{g} ]\!] \vdash \ldots$$

receive effect channel → $\mathsf{F} \bullet \mathsf{g}$

send effect channel → $\mathsf{g}$

$$[\![ \Gamma \vdash \mathbf{run}\ M : \tau ]\!]_r$$

$$= \nu eff\ .\ ([\![ \Gamma \vdash \mathrm{M} : \mathbf{T}\ \mathsf{F}\ \tau ]\!]_r^{eff} \mid H(\underline{eff}))$$

# Embedding

$$(\!|\, \Gamma \vdash \; <M> : \mathsf{T}\; \varnothing\; \tau \,|\!)^{ei,eo}_{r} \quad = \quad ei?(c).\, (\!|\, M \,|\!)_r\, .\underline{eo}!\langle c \rangle$$

where $\forall \mathsf{g}\,.\quad [\![\, \Gamma \,]\!];\;\; r : ![\![\tau]\!],\; ei : ?[\![\, \mathsf{g} \,]\!],\; \underline{eo} : ![\![\, \mathsf{g} \,]\!] \vdash ei?(c).\, (\!|\, M \,|\!)_r\, .\underline{eo}!\langle c \rangle$

$$(\!|\, \Gamma \vdash \mathbf{let}\; x \Leftarrow M \;\mathbf{in}\; N : \mathsf{T}\; (\mathsf{F} \bullet \mathsf{G})\; \tau \,|\!)^{ei,eo}_{r} \quad =$$

$$\nu\; q,\; a\, .\; \left( (\!|\, M \,|\!)^{ei,\, a}_{q} \;\middle|\; \underline{q}?(x).(\!|\, N \,|\!)^{a,\, eo}_{r} \right)$$

where $\forall \mathsf{h}\,.\quad q : ![\![\sigma]\!],\;\; ei : ?[\![\, \mathsf{F} \bullet \mathsf{h} \,]\!],\; \underline{a} : ![\![\, \mathsf{h} \,]\!] \vdash (\!|\, M \,|\!)^{ei,\, a}_{q}$

$\mathsf{h} \to \mathsf{G} \bullet \mathsf{h}'$

$\forall \mathsf{h}'\,.\quad x : [\![\sigma]\!];\; r : ![\![\tau]\!],\; a : ?[\![\, \mathsf{G} \bullet \mathsf{h}' \,]\!],\;\; \underline{eo} : ![\![\, \mathsf{h}' \,]\!] \vdash (\!|\, N \,|\!)^{a,\, eo}_{r}$

# Embedding

$$(\!|\ \Gamma \vdash\ <M>\ :\ \mathsf{T}\ \varnothing\ \tau\ |\!)^{ei,eo}_{r} \quad = \quad ei?(c).\ (\!|\ M\ |\!)_r\ .\underline{eo}!\langle c\rangle$$

where $\forall \mathsf{g}\ .\ \ [\![\ \Gamma\ ]\!];\ \ r : ![\![\tau]\!],\ ei : ?[\![\ \mathsf{g}\ ]\!],\ \underline{eo} : ![\![\ \mathsf{g}\ ]\!] \vdash ei?(c).\ (\!|\ M\ |\!)_r\ .\underline{eo}!\langle c\rangle$

$$(\!|\ \Gamma \vdash \mathbf{let}\ x \Leftarrow M\ \mathbf{in}\ N : \mathsf{T}\ (\mathsf{F} \bullet \mathsf{G})\ \tau\ |\!)^{ei,eo}_{r} \quad =$$

$$\nu\ q,\ a\ .\ \left((\!|\ M\ |\!)^{ei,\,a}_{q}\ \mid\ \underline{q}?(x).(\!|\ N\ |\!)^{a,\,eo}_{r}\right)$$

where $\forall \mathsf{h}\ .\ \ q : ![\![\sigma]\!],\ \ ei : ?[\![\ \mathsf{F} \bullet \mathsf{G} \bullet \mathsf{h}'\ ]\!],\ \underline{a} : ![\![\ \mathsf{G} \bullet \mathsf{h}'\ ]\!] \vdash (\!|\ M\ |\!)^{ei,\,a}_{q}$

$\mathsf{h} \rightarrow \mathsf{G} \bullet \mathsf{h}'$

$\forall \mathsf{h}'\ .\ \ x : [\![\sigma]\!];\ r : ![\![\tau]\!],\ a : ?[\![\ \mathsf{G} \bullet \mathsf{h}'\ ]\!],\ \ \underline{eo} : ![\![\ \mathsf{h}'\ ]\!] \vdash (\!|\ N\ |\!)^{a,\,eo}_{r}$

# Example

$$[\![ \ \Gamma \ ]\!], r : ![\![\mathrm{int}]\!], \ \mathit{eff} : [\![ \ [\mathrm{get}\ \mathbb{Z}, \mathrm{put}\ \mathbb{Z}] \ ]\!] \vdash$$

$$[\![ \ \Gamma \vdash \mathbf{let}\ x \Leftarrow \mathsf{get}\ \mathbf{in}\ \mathsf{put}\ (x+1) : \mathsf{T}\ [\mathrm{get}\ \mathbb{Z}, \mathrm{put}\ \mathbb{Z}]\ \mathbb{Z} \ ]\!]_r^{\mathit{eff}}$$

$$[\![ \ \Gamma \vdash \mathbf{run}\ M : \tau \ ]\!]_r = \nu \mathit{eff} \,.\, ([\![ \ \Gamma \vdash \ M : \mathsf{T}\ \mathsf{F}\ \tau ]\!]_r^{\mathit{eff}} \mid \mathrm{Var}(\mathit{eff}, 0))$$

## Soundness

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

$$[\![\Gamma]\!]; (r :! [\![\tau]\!].\mathbf{end}, e : [\![F]\!]) \vdash [\![M]\!]^e_r \approx [\![N]\!]^e_r$$

$\mathbf{let}\ x \Leftarrow M\ \mathbf{in}$ <$x$> $=$ $M$ (left unit)

$\mathbf{let}\ x \Leftarrow$ <$v$> $\mathbf{in}\ M$ $=$ $M[v/x]$ (right unit)

$\mathbf{let}\ x \Leftarrow M\ \mathbf{in}\ (\mathbf{let}\ y \Leftarrow N_1\ \mathbf{in}\ N_2) =$ (associativity)

$\mathbf{let}\ y \Leftarrow (\mathbf{let}\ x \Leftarrow M\ \mathbf{in}\ N_1)\ \mathbf{in}\ N_2$ $[\mathit{if}\ x \# N_1]$

## Completeness

$$[\![\ \Gamma\ ]\!], r : ![\![\tau]\!],\ \mathit{eff} : [\![\ \mathsf{F}\ ]\!] \vdash [\![M]\!]^{\mathit{eff}}_r \approx [\![N]\!]^{\mathit{eff}}_r$$

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

# Application

*Use session-π as intermediate language*

- Effect-informed optimisations, e.g. implicit parallelism

if $\Gamma \vdash M : \mathsf{T}\ \varnothing\ \sigma$ and $\Gamma \vdash N : \mathsf{T}\ \mathrm{F}\ \mathrm{t}$

then $$(\!|\, \mathbf{let}\, x \leftarrow M\, \mathbf{in}\, (\mathbf{let}\, y \leftarrow N\, \mathbf{in}\, P) \,|\!)_r^{\mathsf{ei},\mathsf{eo}}$$
$$= \nu\, q, s, \mathsf{eb}.\ ([\![M]\!]_q \mid (\!|\, N \,|\!)_s^{\mathsf{ei},\mathsf{eb}} \mid \overline{q}?(x).\overline{s}?(y).(\!|\, P \,|\!)_r^{\mathsf{eb},\mathsf{eo}})$$

- Semantics of *concurrent effects*
  - e.g., *non-interference, atomicity via sessions*

# Effects as sessions (summary)

- Sessions and session types expressive enough to encode effects with a **causal** effect system
  - Per effect notion [e.g., state, counting, I/O]:

    effect mapping, handler, encoding operations
- Extended to **case** and **fix** effect-control-flow operator
- Set-based effect systems recovered by transforming causal

*Details see dorchard.co.uk:*
"Using session types as an effect system" (Orchard, Yoshida, PLACES 2015)

# Effects as sessions

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

# Sessions as effects?

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

- Reuse existing effect-system approaches:
  - *Embedding effect systems in Haskell* (Orchard, Petricek, 2014)
- Session types for existing libraries (e.g., CloudHaskell)

# Send/receive as effects

$$\text{(recv)}\ \frac{\Gamma, x : A \ ; \Delta, c : S \vdash P}{\Gamma \ ; \Delta, c : ?[A].S \vdash c?(x).P} \qquad \text{(send)}\ \frac{\Gamma \ ; \Delta, c : S \vdash P \quad \Gamma; . \vdash V : A}{\Gamma \ ; \Delta, c : ![A].S \vdash c?\langle V \rangle.P}$$

- Composition by prefixing

$$?[\text{-}] : \text{ty} \rightarrow (\text{S} \rightarrow \text{S}) \qquad ?[\text{-}]\ \text{t}\ \text{S} = ?[\text{t}].\text{S}$$

$$![\text{-}] : \text{ty} \rightarrow (\text{S} \rightarrow \text{S}) \qquad ![\text{-}]\ \text{t}\ \text{S} = ![\text{t}].\text{S}$$

$$\textbf{end} : \text{S}$$

- Equivalent to a list over token S (cf. different lists in Prolog)

$$:? \ : \ \text{ty} \rightarrow \text{S} \qquad {+}{+} : [\text{S}] \rightarrow [\text{S}] \rightarrow [\text{S}]$$

$$:! \ : \ \text{ty} \rightarrow \text{S} \qquad [\,] : [\text{S}]$$

# Send/receive as effects

$$\text{(recv)}\ \frac{\Gamma, x : A\ ;\ \Delta, c : S \vdash\ P}{\Gamma\ ;\ \Delta, c : ?[A].S \vdash\ c?(x).P} \qquad \text{(send)}\ \frac{\Gamma\ ;\ \Delta, c : S \vdash\ P \qquad \Gamma;\ . \vdash V : A}{\Gamma\ ;\ \Delta, c : ![A].S \vdash\ c?\langle V \rangle.P}$$

- Decompose environment Δ

effects (List {c :? ***t***, c :! ***t***}, ++, [ ])

```
recv :: Chan c t → T '[c :? t] t
send :: Chan c t → t → T '[c :! t] ()
```

# Effect-indexed monads

```
class Effect (t :: ef → * → *) where
 type Unit t
 type Plus t f g

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

gives

$$\frac{\Gamma \vdash M : \texttt{T f a} \qquad \Gamma, x : \texttt{a} \vdash N : \texttt{T g b}}{\Gamma \vdash \texttt{do}\ x \leftarrow M;\ N : \texttt{T (Plus T f g) b}}$$

cf.

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

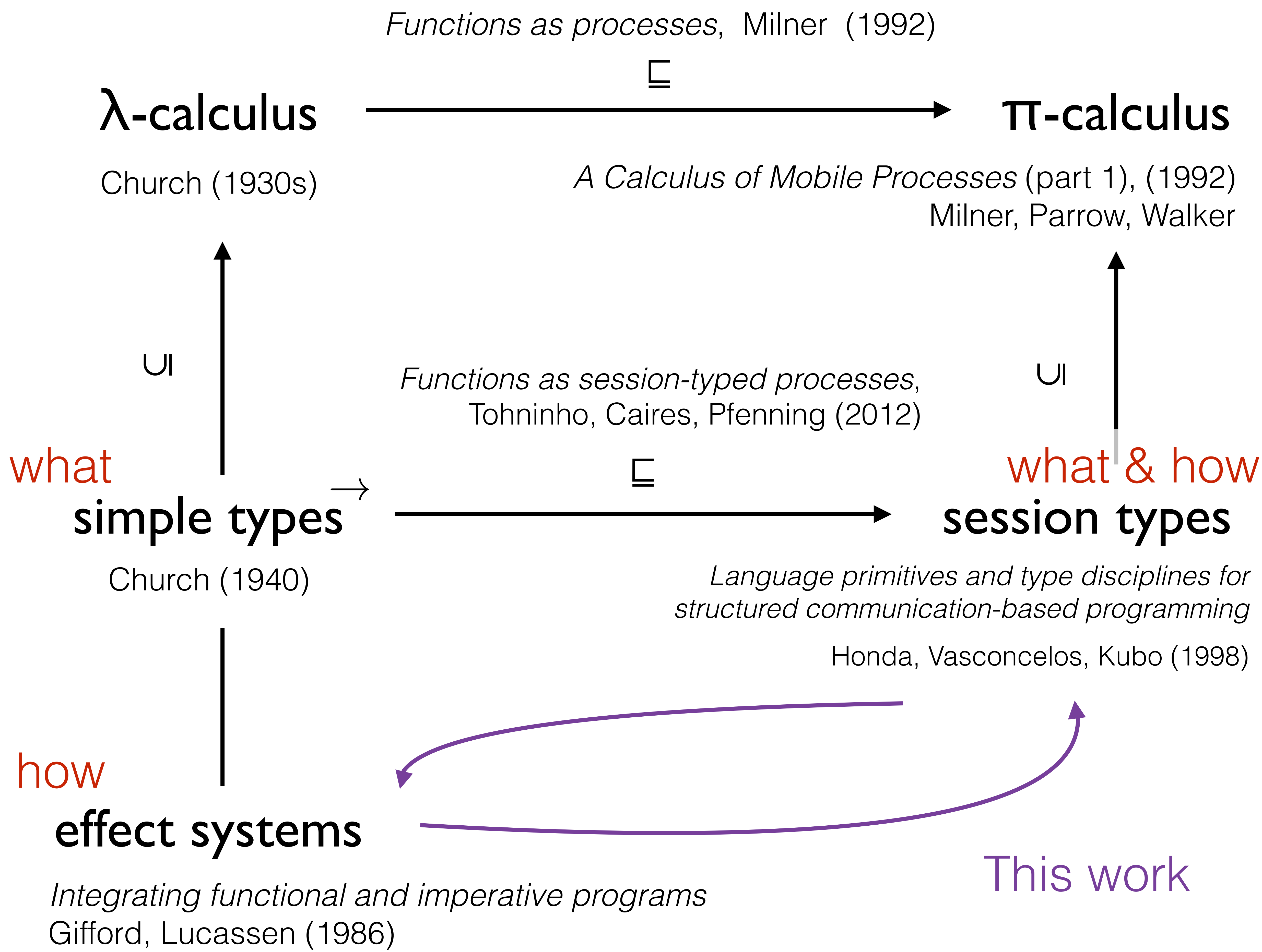
[“The semantic marriage of effects and monads”, Orchard, Petricek, Mycroft (2014)]
[“Parametric effect monads”, Katsumata (2014)]
[“Embedding effect systems in Haskell”, Orchard, Petricek (2014)]

# Session-indexed monads

```
data S a = C :? a | C :! a

data Session (s :: [S *]) a = …

instance Session where
 type Unit m       = ‘[]
 type Plus m s t   = s :++ t

 return :: a → m (Unit m) a
 (>>=) :: m s a → (a → m t b) → m (Plus m s t) b


recv :: Chan c t → Session '[c :? a] t
send :: Chan c t → t → Session '[c :! t] ()
```

# Duality and par

```
par :: Session s () → Session t ()
                    → Session (s :++ t) PId

new :: Duality c s =>
         (Chan c -> Session s a) → Session (Rem c s) a
```



Functions as processes, Milner (1992)
⊑
λ-calculus
π-calculus
Church (1930s)
A Calculus of Mobile Processes (part 1), (1992)
Milner, Parrow, Walker
⊆
⊆
Functions as session-typed processes,
Tohninho, Caires, Pfenning (2012)
what
what & how
→
⊑
simple types
session types
Church (1940)
Language primitives and type disciplines for
structured communication-based programming
Honda, Vasconcelos, Kubo (1998)
how
effect systems
This work
Integrating functional and imperative programs
Gifford, Lucassen (1986)

# Conclusion

## Effects into sessions

- Shows the expressive power of session types
- Incorporate effect information into specifications (e.g. Scribble)
- Use pi-calculus as intermediate language

## Sessions into effects

- Embed session types into existing languages
- Shows the expressive power of effect typing
  - "*Type & Effect system: Behaviours for concurrency*" Amtoft, Nielson, Nielson 1999

one

the tale of ~~two~~ type system~~s~~ ?

Thanks!!!