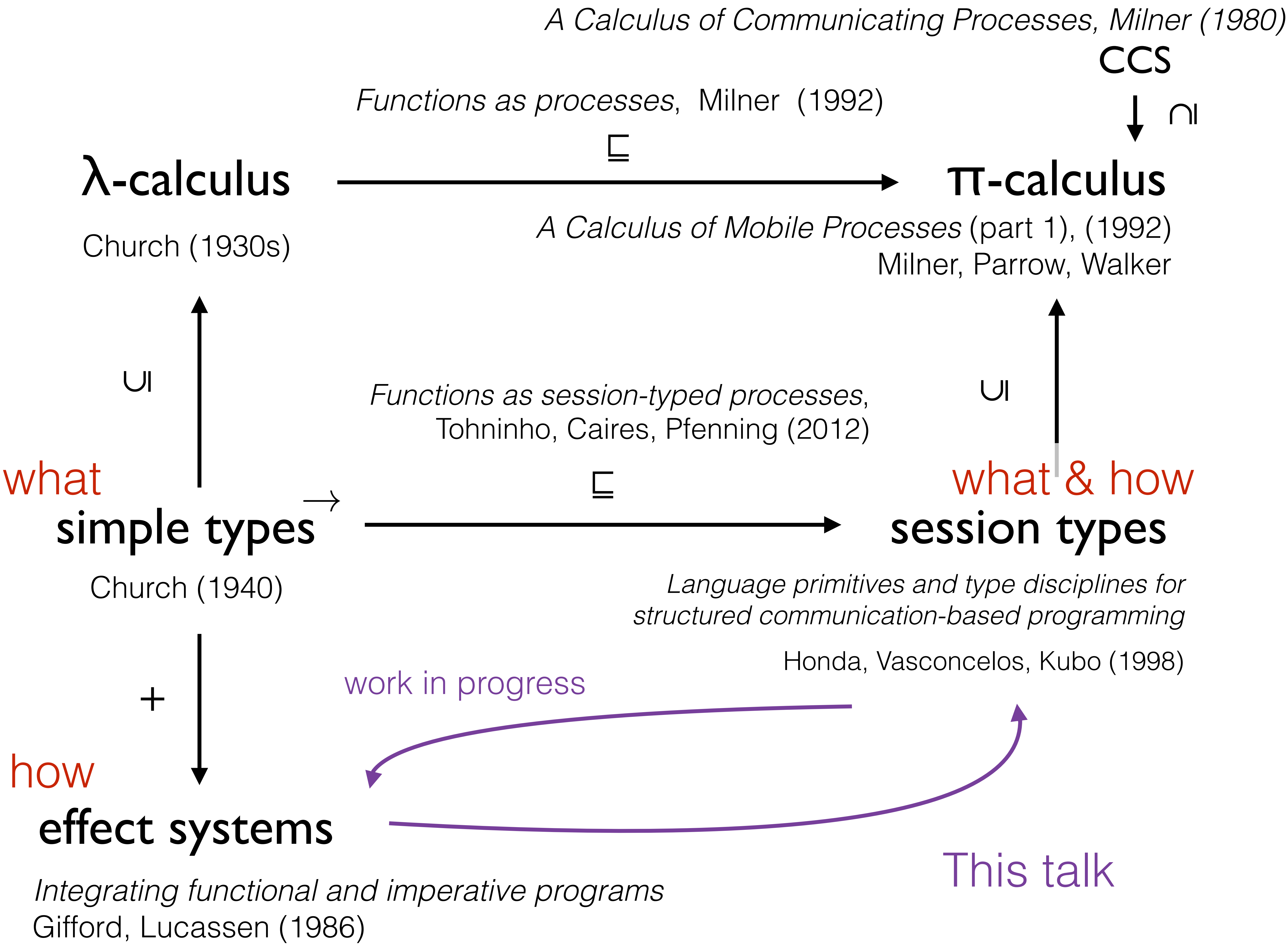
Effects in a pi -
using session types as an effect system

Dominic Orchard, Nobuko Yoshida

Imperial College London

dorchard.co.uk


A Calculus of Communicating Processes, Milner (1980)
CCS
⊇
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)
work in progress
+
how
effect systems
This talk
Integrating functional and imperative programs
Gifford, Lucassen (1986)

π-calculus recap

$c!\langle V\rangle.P$	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

$$(c?(x).P \ \mid\ c!\langle V\rangle.Q) \ \rightarrow\ (P[V/x] \mid Q)$$ (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

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

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

dual end-point

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

(β reduction)

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

(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*

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

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

(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*

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

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

(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)

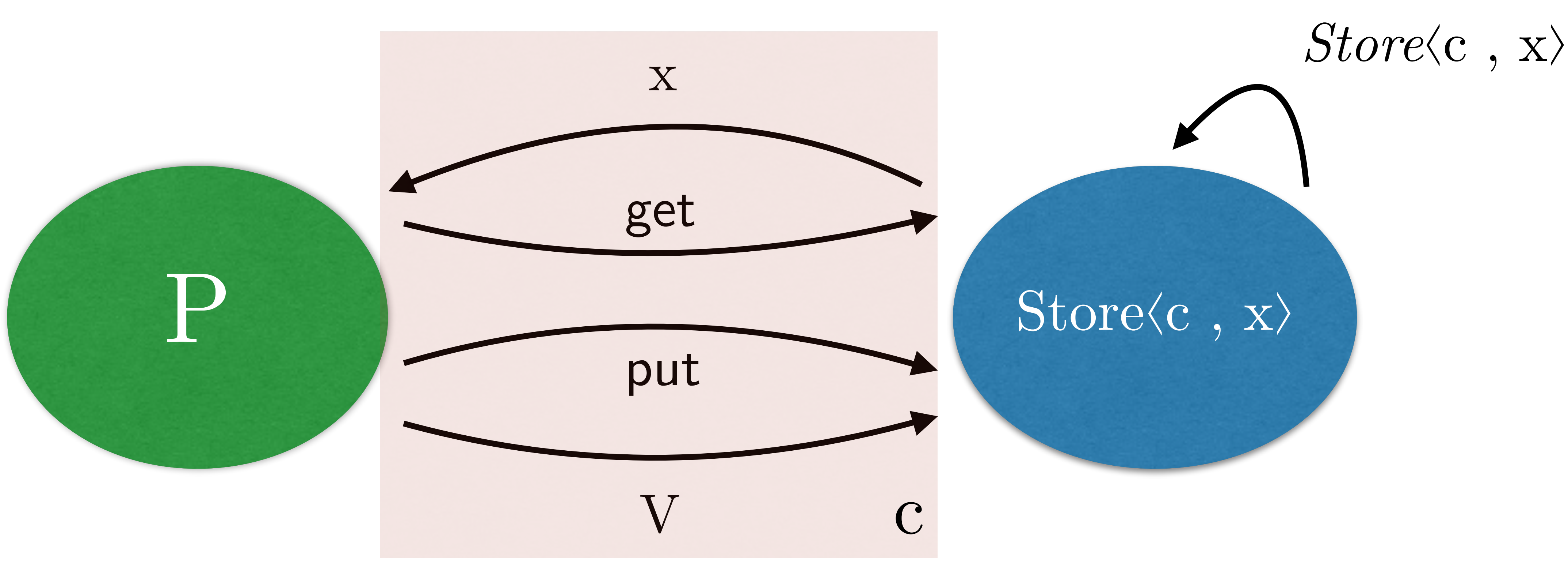
(branch)
$$\frac{\Gamma; \Delta,\ c : S_0 \vdash P_0 \quad \ldots \quad \Gamma; \Delta,\ c : S_n \vdash P_n}{\Gamma\ ;\ \Delta,\ c : \&[l_0 : S_0 \ldots\ l_n : S_n] \vdash c \triangleright \{l_0 : P_0 \ldots\ l_n : P_n\}}$$

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

Duality:

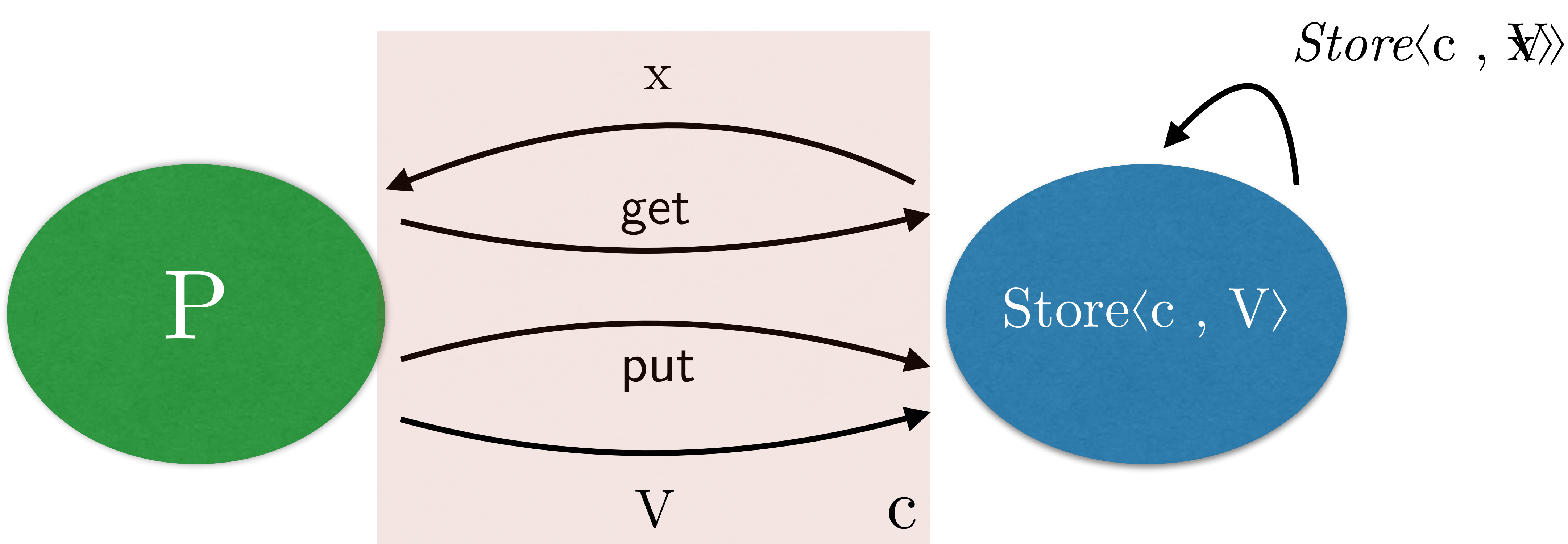
$$\underline{\&[l_0 : S_0 \ldots\ l_n : S_n]} = \oplus[l_0 : \underline{S_0} \ldots\ l_n : \underline{S_n}]$$
$$\underline{\oplus[l_0 : S_0 \ldots\ l_n : S_n]} = \&[l_0 : \underline{S_0} \ldots\ l_n : \underline{S_n}]$$

Effects in a π : *Variable agent*



Effects in a π : *Variable agent*

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



Effects in a π : *Variable agent*

Server

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

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

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

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

describes effect interaction

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

Effect system

monoid (F,•,∅)

abs $\dfrac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x.M : \sigma \to \tau}$ app $\dfrac{\Gamma \vdash M : \sigma \to \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash M\ N : \tau}$ var $\dfrac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}$

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

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

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

$$\frac{\Gamma, x : \sigma \vdash M : \tau, \mathsf{F}}{\Gamma \vdash \lambda x.M : \sigma \xrightarrow{\mathsf{F}} \tau, \varnothing} \qquad \frac{\Gamma \vdash M : \sigma \xrightarrow{\mathsf{H}} \tau, \mathsf{F} \qquad \Gamma \vdash N : \sigma, \mathsf{G}}{\Gamma \vdash M\ N : \tau, \mathsf{F} \bullet \mathsf{G} \bullet \mathsf{H}}$$

Effect system for state

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

Effect operations: $\text{put} : \tau \rightarrow \mathsf{T}\ [\text{put}\ \tau]\ ()$

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

e.g. increment store

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

cf. session-typed π-calculus version

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

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

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

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

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

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

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

Embedding

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

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

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

both using 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

send effect channel

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

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

Embedding

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

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

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

$$\nu\ q,\ a\ .\ (\llparenthesis\ M\ \rrparenthesis^{ei,\,a}_{q}\ \ |\ \underline{q}?(x).\llparenthesis\ N\ \rrparenthesis^{a,\,eo}_{r}\ )$$

where $\forall {\color{orange}\mathsf{h}}\ .\qquad q : ![\![\sigma]\!],\ \ ei : ?[\![\ {\color{orange}\mathsf{F} \bullet \mathsf{h}}\ ]\!],\ \underline{a} : ![\![\ {\color{orange}\mathsf{h}}\ ]\!] \vdash \llparenthesis\ M\ \rrparenthesis^{ei,\,a}_{q}$

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

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

Embedding

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

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

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

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

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

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

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

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

Soundness

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

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

$\textbf{let}\ x \Leftarrow M\ \textbf{in}\ \langle x \rangle \ = \ M$ (left unit)

$\textbf{let}\ x \Leftarrow \langle v \rangle\ \textbf{in}\ M \ = \ M[v/x]$ (right unit)

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

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

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 $(\!|\, \mathbf{let}\, x \leftarrow M\, \mathbf{in}\, (\mathbf{let}\, y \leftarrow N\, \mathbf{in}\, P)\, |\!)_r^{\mathsf{ei,eo}}$

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

- 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)

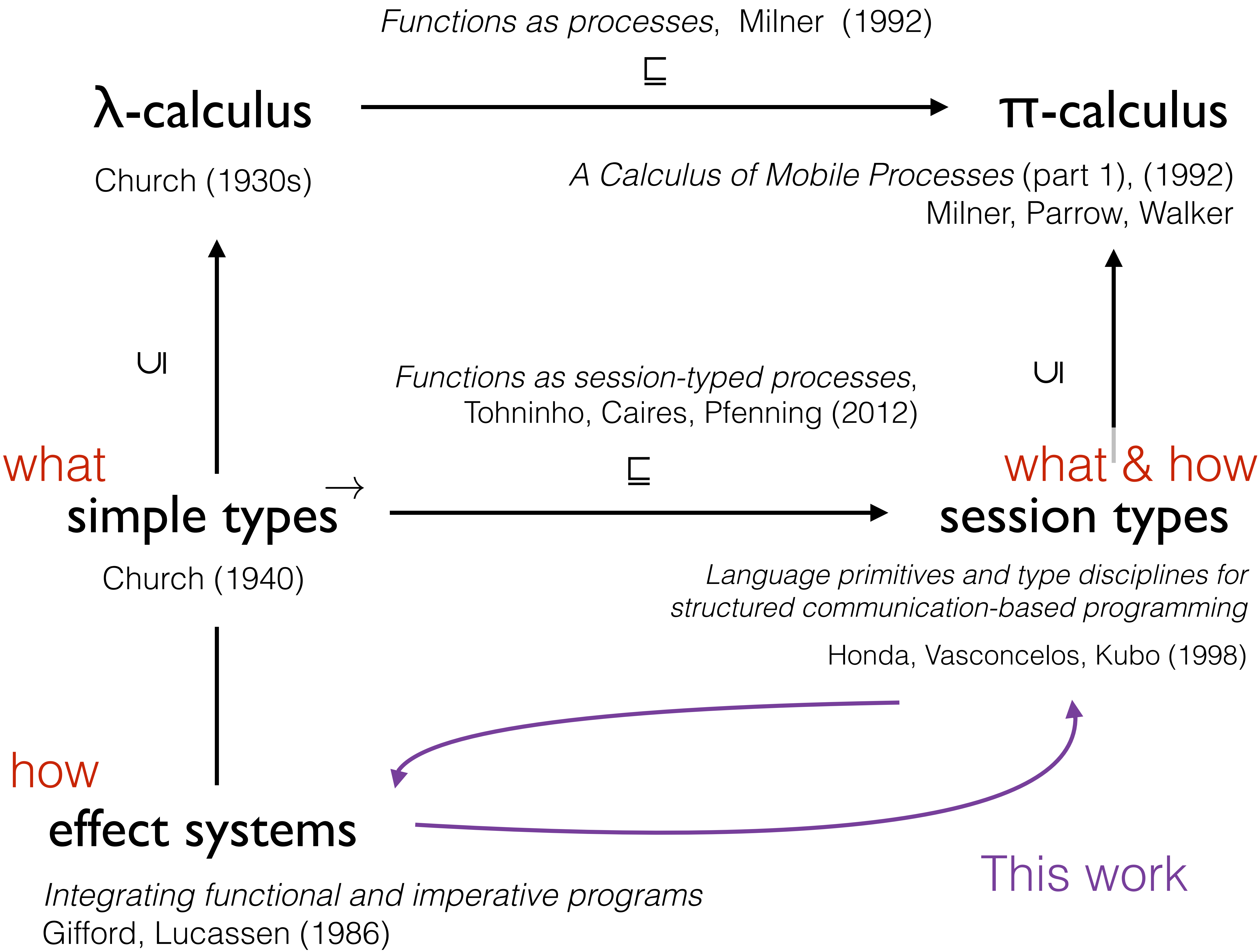
Effects as sessions

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

Sessions as effects?

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

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



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 (work in progress)

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

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

Thanks!!!

Other things I do…

Theory

- **Coeffects [dual to effects, contextual effects]** $\forall i.\ \Box_i$

$$\Box_{(i+j)}\ A \rightarrow \Box_i \Box_j\ A \qquad \Box_0 A \rightarrow A \qquad \Box_i(A \rightarrow B) \rightarrow \Box_i A \rightarrow \Box_i B$$

[thesis,ICALP'13,ICFP'14] with Alan Mycroft, Tomas Petricek
[w.i.p] with Marco Gaboardi, Shinya Katsumata

- **Models/analysis of timed, communicating processes [for music!]**

[FARM'14,w.i.p] with Sam Aaron

Applications

- **Haskell type system and compiler**

[thesis,FLOPS'10,IFL'12,ICFP'13,Haskell'14]
with Alan Mycroft, Tomas Petricek, Neal Glew, Leaf Petersen, Tom Schrijvers

- **Languages and tools for computational science**

SSI Fellowship (2015) [WRT'13,ICCS'13,ICCS'14] with Andy Rice