Dependent Session Types for the Higher-Order π-calculus

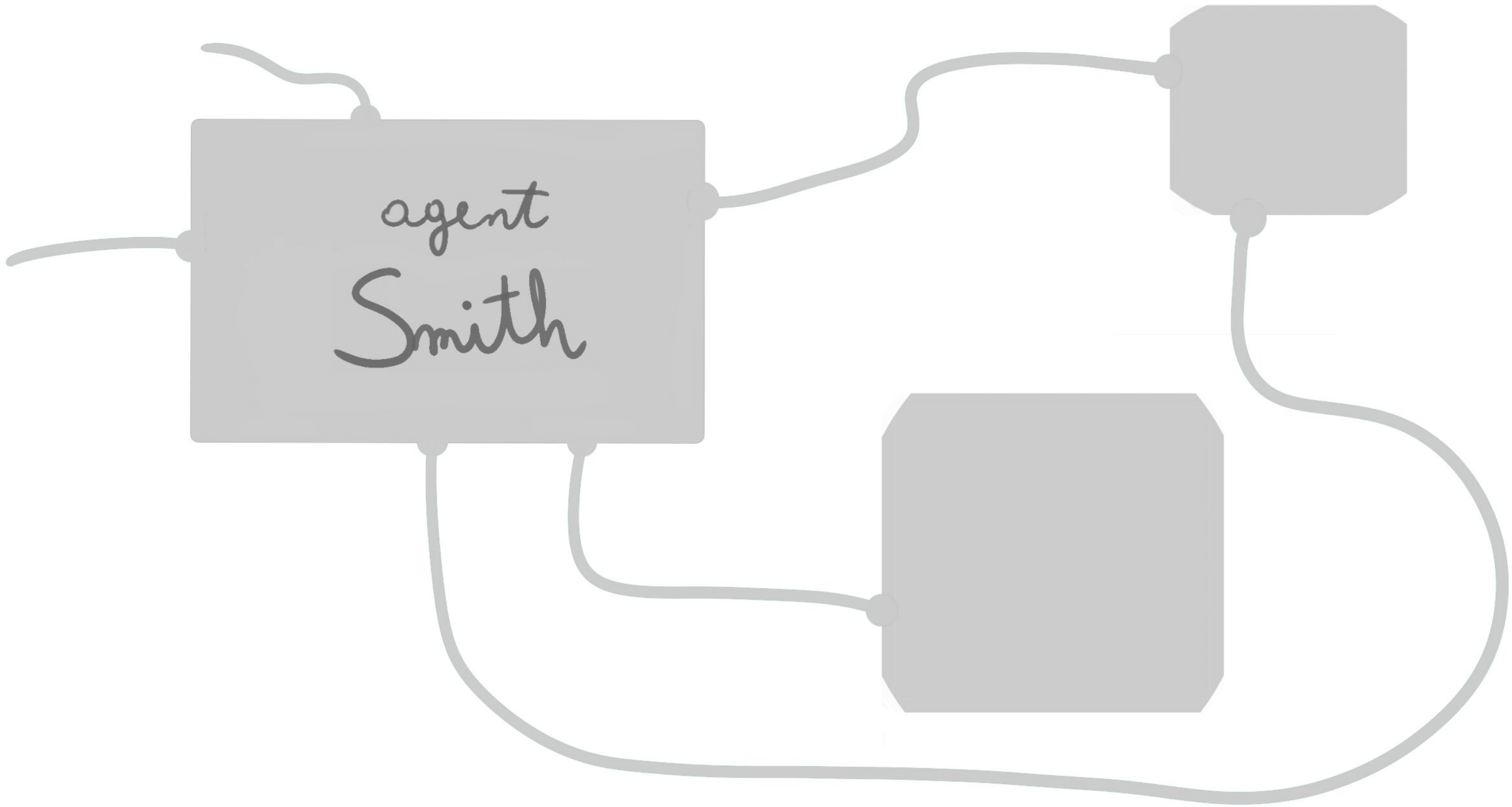
A work in progress by

F. Ferreira
D. Ruoppolo
N. Yoshida

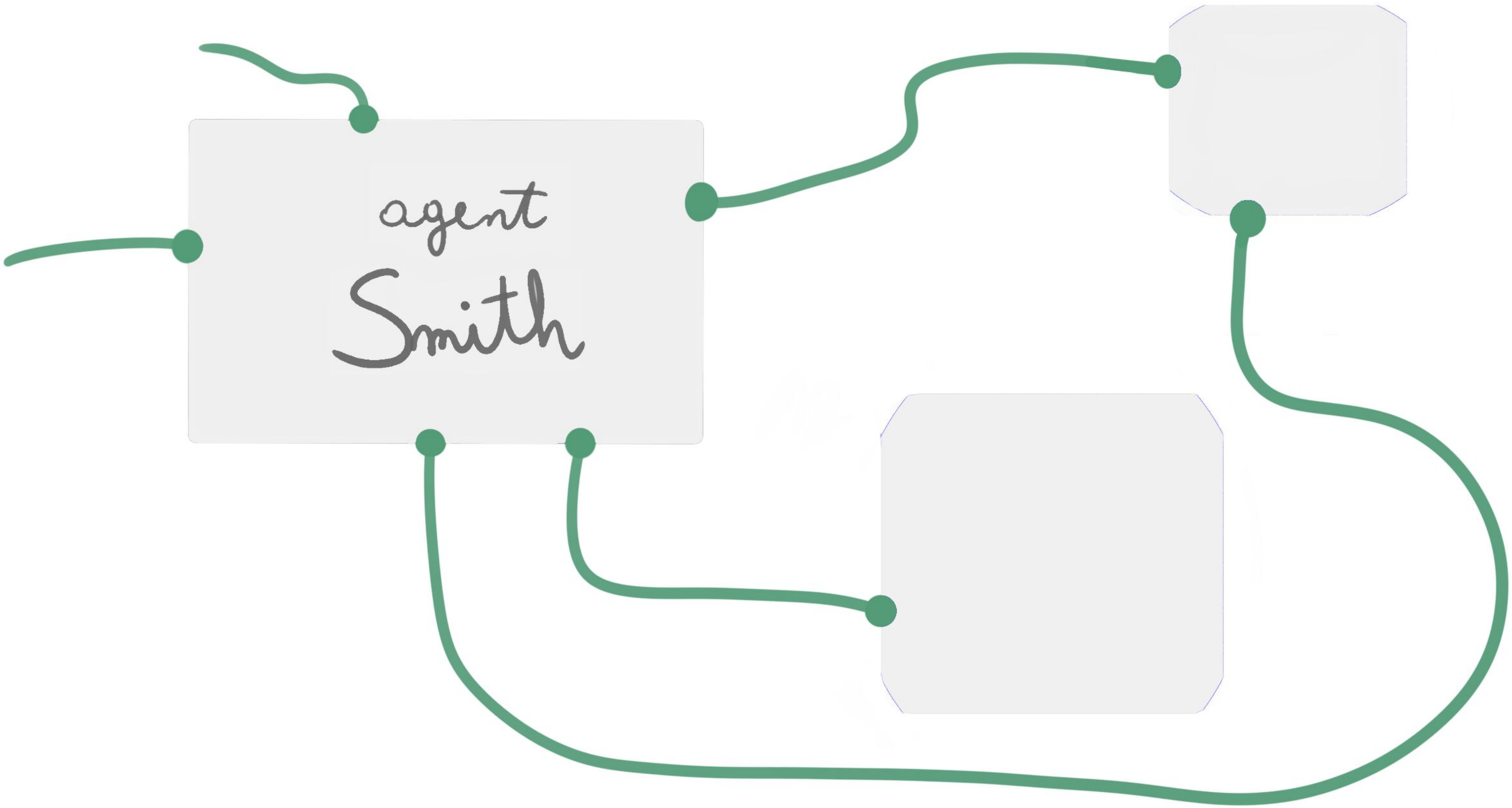
MRG group

Imperial College London

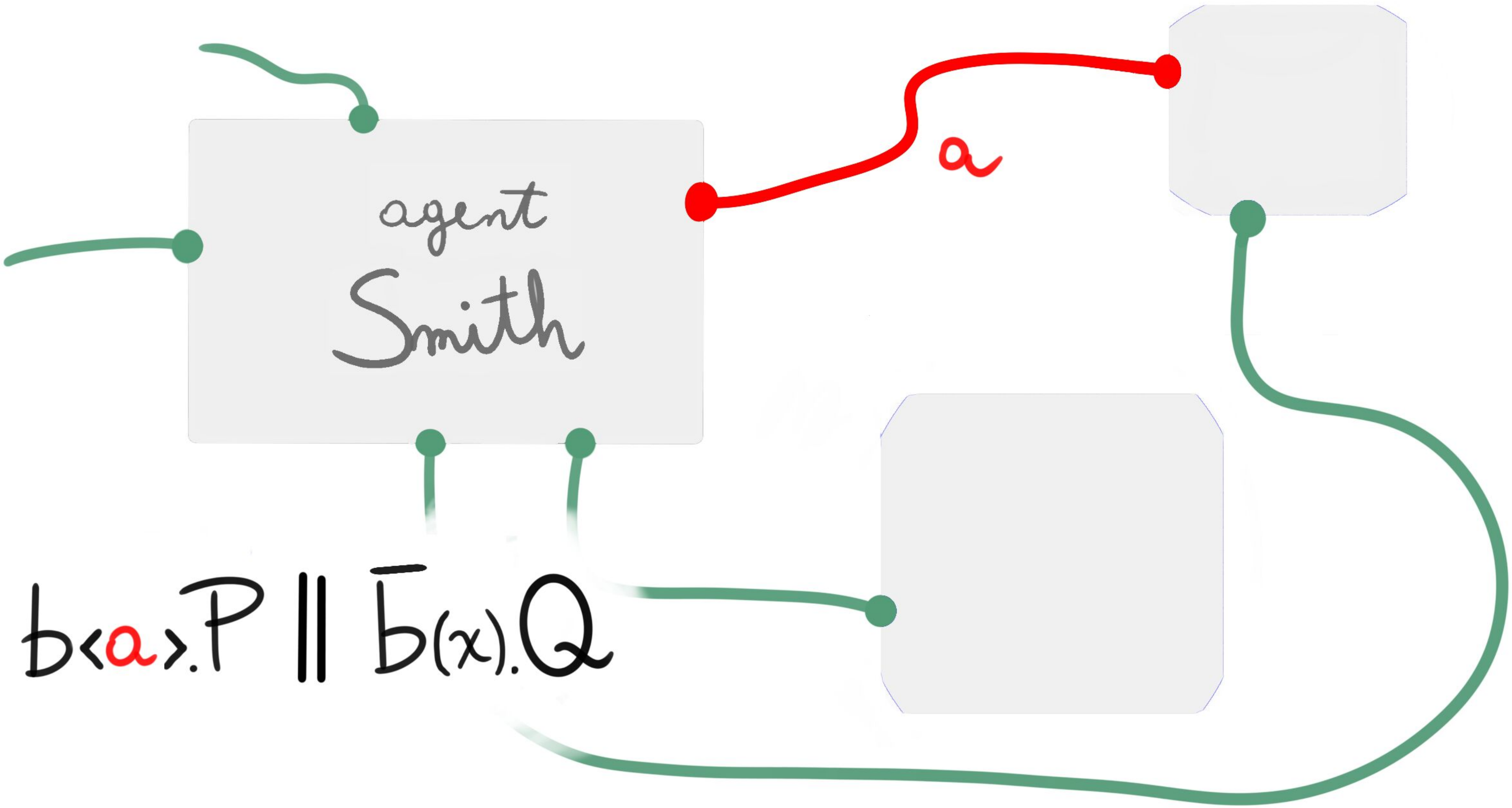
MILNER's π-calculus



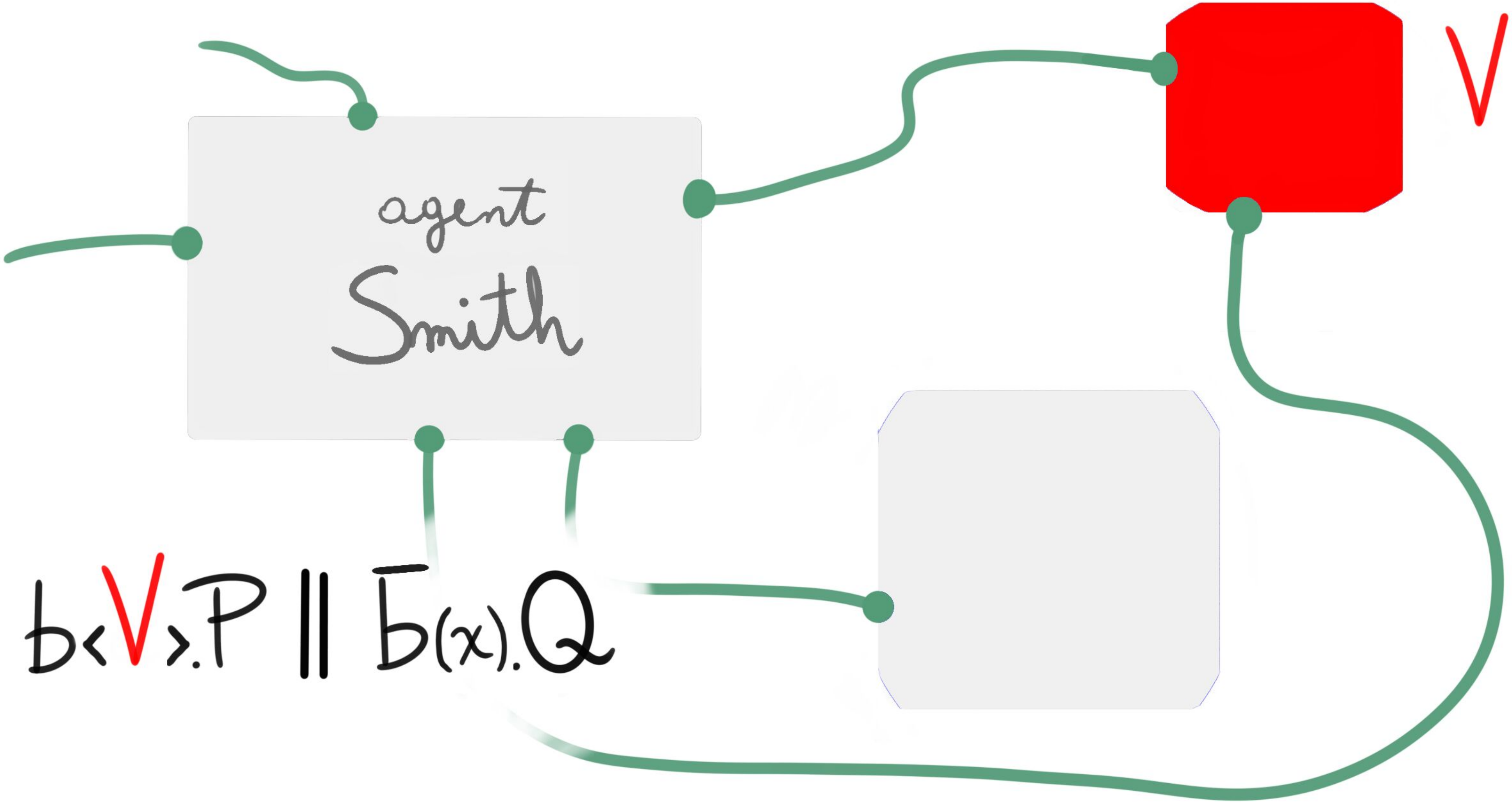
MILNER's π-calculus



MILNER's π-calculus



Milner's HOπ-calculus



Why Dependent Types?

$$\vdash P : a$$

- Process of behaviour a
- Functional program of type a
- Proof certificate of the property a

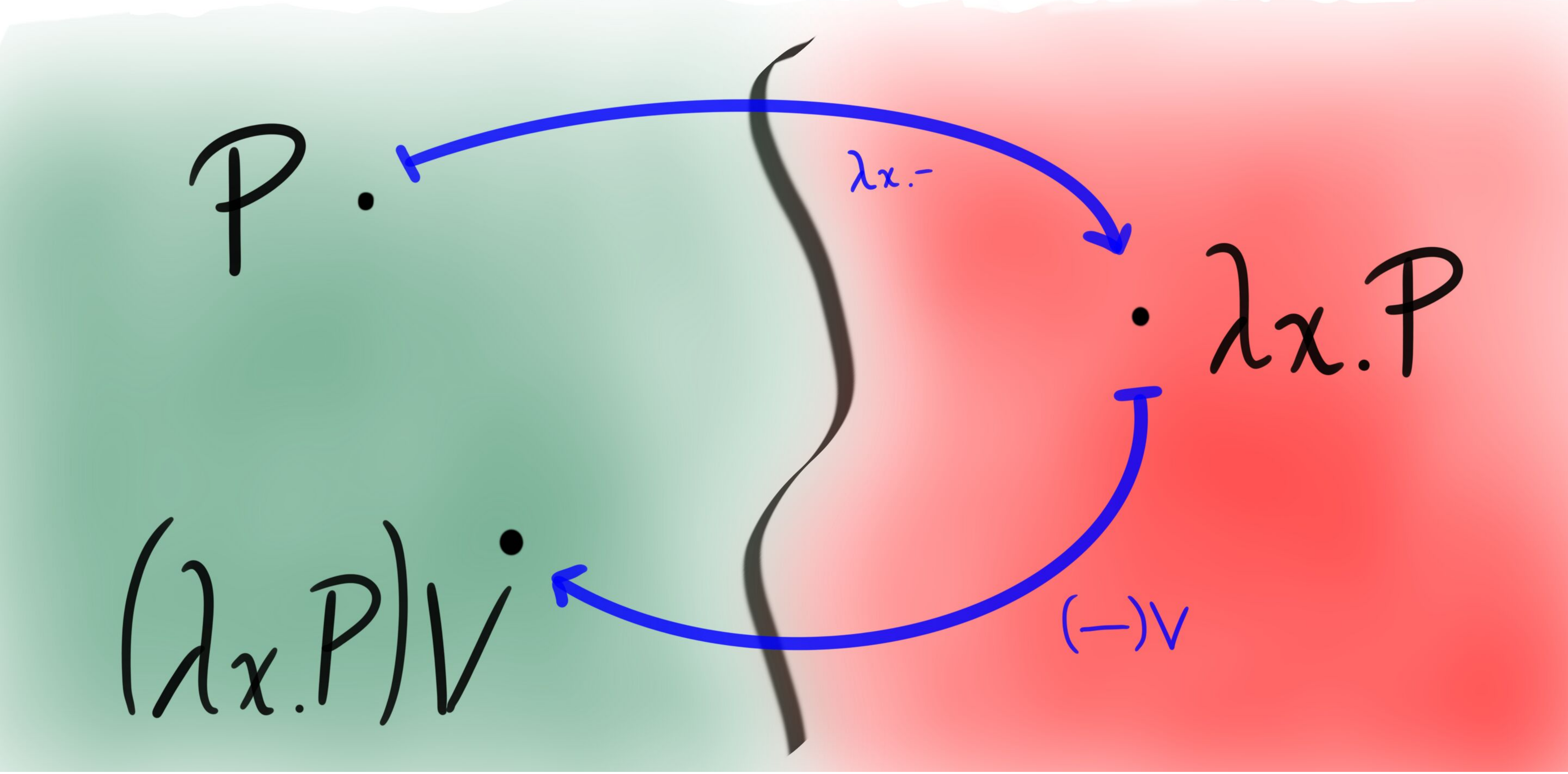
Why Dependent Types?

$$\frac{\vdash P : \beta \qquad \alpha \simeq \beta}{\vdash P : \alpha}$$

Some form of STATICALLY CHECKABLE computation

- Process of behaviour α
- Functional program of type α
- Proof certificate of the property α

Milner-Sangiorgi's HOπ-calculus



~~Milner-Sangiorgi's~~ HOπ-calculus

$$\lambda x_1 . \lambda x_2 . \ldots \lambda x_n . P$$

~~MILNER-SANGIORGI's~~ HOπ-calculus

$$\lambda x_1.\lambda x_2.\dots\lambda x_n.P \ :$$

$$\Pi x_1{:}a_1.\dots\Pi x_n{:}a_n.a$$

Syntax: object terms

$$
\begin{aligned}
u &::= x \mid s \mid a \mid \overline{a} \\
V &::= u \mid * \mid \lambda x{:}\sigma.P \mid \langle P, P\rangle_{\Sigma x:\tau.\sigma} \\
P &::= V \mid PP \mid pr_1 P \mid pr_2 P \mid \\
&\quad\; 0 \mid P \| P \mid \\
&\quad\; u(x{:}\sigma).P \mid u\langle V\rangle.P \mid (\nu a{:}\delta)P \\
&\quad\; \mathrm{accept}\; u(x{:}\sigma).P \mid \mathrm{request}\; u\langle a\rangle.P \mid (\nu s{:}\#\delta)P
\end{aligned}
$$

Syntax: type terms

$$\sigma ::= \tilde{\tau} \mid \delta \mid \# \delta$$

$$\delta ::= \mathsf{end} \mid\ !\sigma.\delta \mid\ ?\sigma.\delta$$

$$\tilde{\tau} ::= X \mid \mathsf{unit} \mid \tau P \mid \Pi x{:}\sigma.\tilde{\tau} \mid \Sigma x{:}\tilde{\tau}.\sigma \mid [\Delta]$$

$$\Delta ::= \emptyset \mid \Delta, u{:}\delta$$

Syntax: Kind terms

$$K ::= \mathsf{Type} \mid \Pi x{:}\sigma.K$$

Syntax: type terms

$$\sigma ::= \tau \mid \delta \mid \#\delta$$

$$\delta ::= \mathrm{end} \mid\ !\sigma.\delta \mid\ ?\sigma.\delta$$

$$\tau ::= X \mid \mathrm{unit} \mid \tau P \mid \Pi x{:}\sigma.\tau \mid \Sigma x{:}\tau.\sigma \mid [\Delta]$$

$$\Delta ::= \emptyset \mid \Delta, u{:}\delta$$

Syntax: Kind terms

$$K ::= \mathrm{Type} \mid \Pi x{:}\sigma.K$$

NO

$$\tau \to \delta$$

Contexts: linearity or not?

$$\Gamma ; \Delta \vdash P : [\Delta]$$

Contexts: linearity or not?

$$\Gamma ; \Delta \vdash P : [\Delta]$$

$$x : \sigma \vdash x : \sigma$$

$$x : !\mathbb{B}!\mathbb{B}.\mathrm{end} \vdash x\langle \mathtt{tt} \rangle . x\langle \mathtt{ff} \rangle . 0 : [x : !\mathbb{B}!\mathbb{B}.\mathrm{end}]$$

Contexts: linearity or not?

$$\Gamma ; \cancel{\Delta} \vdash P : [\Delta]$$

$$X : \mathrm{unit} \to \mathrm{Type},\ y : \mathrm{unit},\ z : X_y,\ a : !(X_y).\mathrm{end},\ b : ?\mathbb{B}.\mathrm{end} \vdash a\langle z \rangle.0 : [a : !(X_y).\mathrm{end}]$$

Contexts: linearity or not?

$$X : \mathrm{unit} \to \mathsf{Type},\ y : \mathrm{unit},\ z : X_y,\ a : !_{\mathrm{unit}}?\mathbb{N}\,!(X_y).\mathrm{end} \vdash a\langle z\rangle.0 : \left[a : !(X_y).\mathrm{end}\right]$$

Contexts : linearity or not ?

(input)

$$\frac{\Gamma, x : \sigma \vdash P : \left[\Delta, u : \boldsymbol{\delta}, \Delta'^{(x:\sigma)}\right] \qquad \Gamma \vdash \left[\Delta, u : ?\sigma.\boldsymbol{\delta}, \Delta'\right] : \textsf{Type}}{\Gamma \vdash u(x : \sigma).P : \left[\Delta, u : ?\sigma.\boldsymbol{\delta}, \Delta'\right]} \; ?\sigma.\boldsymbol{\delta} \preceq \Gamma(u)$$

$$X : \mathrm{unit} \to \mathrm{Type},\; y : \mathrm{unit},\; z : X_y,\; a : !_{\mathrm{unit}}?\mathbb{N}\,!(X_y).\mathrm{end} \vdash a\langle z\rangle.0 : \left[a : !(X_y).\mathrm{end}\right]$$

Balance for free !

(chan-Ctx)

$$\frac{\Gamma \vdash \boldsymbol{\delta} : \texttt{SessionType}}{\Gamma, e : \boldsymbol{\delta} \ \ \mathsf{wf}} \ e, \overline{e} \notin \mathsf{dom}(\Gamma)$$

(dualChan-Ctx)

$$\frac{\Gamma \vdash \boldsymbol{\delta} : \texttt{SessionType}}{\Gamma, \overline{e} : \overline{\boldsymbol{\delta}} \ \ \mathsf{wf}} \ \Gamma(e) = \boldsymbol{\delta} \ \text{ and } \ \overline{e} \notin \mathsf{dom}(\Gamma)$$

(Process-T)

$$\frac{\Gamma \ \mathsf{wf} \qquad \Delta \ \mathsf{wf}}{\Gamma \vdash \big[\,\Delta\,\big] : \texttt{TargetType}} \ \mathsf{dom}(\Delta) \subseteq \mathsf{dom}(\Gamma) \ \text{ and } \ \Delta(u) \preceq \Gamma(u) \ \ \forall\, u \in \mathsf{dom}(\Delta)$$

Balance for free!

(chan-Ctx)

$$\frac{\Gamma \vdash \boldsymbol{\delta} : \texttt{SessionType}}{\Gamma, e : \boldsymbol{\delta}\ \ \mathsf{wf}}\ e, \overline{e} \notin \mathsf{dom}(\Gamma)$$

(dualChan-Ctx)

$$\frac{\Gamma \vdash \boldsymbol{\delta} : \texttt{SessionType}}{\Gamma, \overline{e} : \overline{\boldsymbol{\delta}}\ \ \mathsf{wf}}\ \Gamma(e) = \boldsymbol{\delta}\ \text{ and }\ \overline{e} \notin \mathsf{dom}(\Gamma)$$

(Process-T)

$$\frac{\Gamma\ \ \mathsf{wf} \qquad \Delta\ \ \mathsf{wf}}{\Gamma \vdash [\,\Delta\,] : \texttt{TargetType}}\ \mathsf{dom}(\Delta) \subseteq \mathsf{dom}(\Gamma)\ \text{ and }\ \Delta(u) \preceq \Gamma(u)\ \forall\, u \in \mathsf{dom}(\Delta)$$

automatically balanced!

Balance for free!

(chan-Ctx)

$$\frac{\Gamma \vdash \boldsymbol{\delta} : \mathtt{SessionType}}{\Gamma, e : \boldsymbol{\delta} \ \mathsf{wf}} \quad e, \overline{e} \notin \mathsf{dom}(\Gamma)$$

(dualChan-Ctx)

$$\frac{\Gamma \vdash \boldsymbol{\delta} : \mathtt{SessionType}}{\Gamma, \overline{e} : \overline{\boldsymbol{\delta}} \ \mathsf{wf}} \quad \Gamma(e) = \boldsymbol{\delta} \ \text{ and } \ \overline{e} \notin \mathsf{dom}(\Gamma)$$

(Process-T)

$$\frac{\Gamma \ \mathsf{wf} \qquad \Delta \ \mathsf{wf}}{\Gamma \vdash \left[\, \Delta \,\right] : \mathtt{TargetType}} \quad \mathsf{dom}(\Delta) \subseteq \mathsf{dom}(\Gamma) \ \text{ and } \ \Delta(u) \preceq \Gamma(u) \ \forall\, u \in \mathsf{dom}(\Delta)$$

automatically balanced!

(paral)

$$\frac{\Gamma \vdash P : [\,\Delta\,] \qquad \Gamma \vdash Q : \left[\,\Delta'\,\right] \qquad \Gamma \vdash \left[\,\Delta, \Delta'\,\right] : \mathtt{Type}}{\Gamma \vdash P \parallel Q : \left[\,\Delta, \Delta'\,\right]}$$

Why Dependent Types?

$$\frac{\vdash P : \beta \qquad \alpha \simeq \beta}{\vdash P : \alpha}$$

Some form of STATICALLY CHECKABLE computation

- Process of behaviour α
- Functional program of type α
- Proof certificate of the property α

Computational equivalence

$$\frac{P \equiv P' \qquad P' \rightarrow Q' \qquad Q' \equiv Q}{P \rightarrow Q}$$

Computational equivalence

$$\frac{P \equiv P' \qquad P' \to Q' \qquad Q' \equiv Q}{P \to Q}$$

(1) $(P \parallel P') \parallel P'' \equiv P \parallel (P' \parallel P'');$

(2) $P \parallel Q \equiv Q \parallel P;$

(3) $\big(\nu a : \boldsymbol{\delta}\big)\big(\nu b : \boldsymbol{\delta'}\big)P \equiv \big(\nu b : \boldsymbol{\delta'}\big)\big(\nu a : \boldsymbol{\delta}\big)P;$

(4) $\big(\nu a : \boldsymbol{\delta}\big)P \parallel Q \equiv \big(\nu a : \boldsymbol{\delta}\big)(P \parallel Q)$ *whenever* $a\ , \overline{a} \notin \mathtt{fe}(Q);$

(5) $P \parallel 0 \equiv P.$

(6) $(\nu a : \boldsymbol{\delta})\, 0 \equiv 0.$

Computational equivalence

$$\frac{P \equiv P' \qquad P' \to Q' \qquad Q' \equiv Q}{P \to Q}$$

(1) $(P \parallel P') \parallel P'' \equiv P \parallel (P' \parallel P'')$;

(2) $P \parallel Q \equiv Q \parallel P$;

(3) $(\nu a : \boldsymbol{\delta})(\nu b : \boldsymbol{\delta'})P \equiv (\nu b : \boldsymbol{\delta'})(\nu a : \boldsymbol{\delta})P$;

(4) $(\nu a : \boldsymbol{\delta})P \parallel Q \equiv (\nu a : \boldsymbol{\delta})(P \parallel Q)$ *whenever* $a\ , \overline{a} \ \notin \mathtt{fe}(Q)$;

(5) $P \parallel 0 \equiv P$.

(6) $(\nu a : \boldsymbol{\delta})\,0 \equiv 0$.

Computational equivalence

$$\frac{P \lesssim P' \qquad P' \to_S Q' \qquad Q' \lesssim Q}{P \to_S Q}$$

(i) $(P \parallel P') \parallel P'' \lesssim P \parallel (P' \parallel P'')$;

(ii) $\big(\nu a : \boldsymbol{\delta}\big)P \parallel Q \lesssim \big(\nu a : \boldsymbol{\delta}\big)(P \parallel Q)$ *whenever* $a \;, \overline{a} \;\notin \mathtt{fe}(Q)$;

(iii) $P \parallel 0 \lesssim P$;

(iv) $0 \parallel P \lesssim P$;

(v) $(\nu a : \boldsymbol{\delta})P \lesssim P$ *whenever* $a \;, \overline{a} \;\notin \mathtt{fe}(P)$.