# Full abstraction in a linear subtyped $\pi$-calculus

Romain Demangeon

(joint work with Kohei Honda)
Queen Mary, University of London

Concurrency Workshop, Dublin, 14/04/2011

# Motivations

## A minimal $\pi$-calculus

- Framework for process logics.
- Simple calculus:
  - Output-asynchronous.
  - Branching inside prefixes.
- A linear-affine type systems
  - Control the use of channels.
- Good expressiveness:
  - Embedding usual programmation paradigms.

## A theory of subtyping

- $T_1 \sqsubseteq T_2$:
  - objects of type $T_1$ can have type $T_2$.
  - $T_2$ is more general than $T_1$.
- Branching defines subtyping.

$\pi^{\{1,\overline{1},\omega\}}$: a linear-affine $\pi$-calculus

Embedding a functional calculus

Embedding a session-based calculus

# Our calculus $\pi^{\{1,\overline{1},\omega\}}$

$a, b, c, u, v, x, y$: names/channels, $l$: labels (branching)

$$P ::= (P \mid P) \mid \mathbf{0} \mid X\langle\widetilde{v}\rangle \mid (\mu X(\widetilde{x}).P)\langle\widetilde{v}\rangle \mid \overline{u} \oplus^m l\langle\widetilde{v}\rangle \mid u\&^m_{i\in I}\{l_i(\widetilde{x}_i).P_i\} \mid (\nu a)\ P$$

## Prefixes

- $\overline{u} \oplus^m l\langle\widetilde{v}\rangle$: selection.
  - Branching (asynchronous) output.
  - Select branch labelled $l$ on channel $u$ with message $\widetilde{v}$.
- $u\&^m_{i\in I}\{l_i(\widetilde{x}_i).P_i\}$: choice.
  - Branching input.
  - Offers branches $(l_i)_{i\in I}$ with continuations $P_i$, binding $\widetilde{x}_i$.

## Modes

$m$ can be:

- $1$: linear: exactly one action.
- $\overline{1}$: affine: at most one action.
- $\omega$: replicated.

# Reductions

$$\overline{u} \oplus^{1/\overline{1}} l_j\langle v\rangle \mid u\&^{1/\overline{1}}{}_{i\in I}\{l_i(x_i).P_i\} \longrightarrow P_j\{v/x_j\}$$

$$\overline{u} \oplus^{\omega} l_j\langle v\rangle \mid u\&^{\omega}{}_{i\in I}\{l_i(x_i).P_i\} \longrightarrow u\&^{\omega}{}_{i\in I}\{l_i(x_i).P_i\} \mid P_j\{v/x_j\}$$

- Reductions happen:
  - in $\mid$ of spectator processes,
  - under $\nu$.
- Structural congruence defined "as usual".

# Typing: prefixes

## Types

$$T \ ::= \ \&^m_{i\in I}\{l_i(\widetilde{T}_i)\} \ | \ \oplus^m_{i\in I}\{l_i(\widetilde{T}_i)\} \ | \ \mu t.T \ | \ t \ | \ \mathsf{uc} \ | \ \mathtt{N} \ | \ \mathtt{B} \ | \ \bigstar$$

- Typing judgments: $\Gamma \vdash_\pi P$

## Prefixes

$$(\textbf{Cho})\frac{(\Gamma, \widetilde{x}_i : \widetilde{T}_i \vdash_\pi P_i)_{i\in I}}{\Gamma, u : \&^m_{i\in I}\{l_i(\widetilde{T}_i)\} \vdash_\pi u\&^m_{i\in I}\{l_i(\widetilde{x}_i) : P_i\}}$$

- $\overline{T}$: dual of $T$.

# Typing: parallel composition

$$(\textbf{Par})\frac{\Gamma_1 \vdash_\pi P_1 \qquad \Gamma_2 \vdash_\pi P_2}{\Gamma_1 \odot \Gamma_2 \vdash_\pi P_1 \mid P_2}$$

- Compatibility $\odot$: way to enforce modes.

# Typing: parallel composition

$$(\textbf{Par})\frac{\Gamma_1 \vdash_\pi P_1 \qquad \Gamma_2 \vdash_\pi P_2}{\Gamma_1 \odot \Gamma_2 \vdash_\pi P_1 \mid P_2}$$

- Compatibility $\odot$: way to enforce modes.
- $T_1 \asymp T_2$:
  - coherence of $(T_1, T_2)$.
  - selections $\subseteq$ choices $\Rightarrow$ "$T_1$ matches $T_2$"
- Linearity: if $T_1 \asymp T_2$ and $\mathtt{mod}(T_1) = \mathtt{mod}(T_2) = 1$ then $(\Gamma_1, u : T_1) \odot (\Gamma_2, u : T_2) = \Gamma_1 \odot \Gamma_2, u : \mathrm{uc}$

uc: cannot be used again.

# Typing: Misc.

## Shortcuts

In the following:

$$u(x).P_1 = u\&^1_{i\in\{1\}}\{l_i(x_i).P_i\} \qquad !u(x) \qquad \overline{u}\langle v\rangle$$

$$\uparrow^m (T) = \oplus^m_{i\in\{1\}}\{l_i(T)\} \qquad \downarrow^m (T)$$

# Typing: Misc.

## Shortcuts

In the following:

$$u(x).P_1 = u\&^1_{i\in\{1\}}\{l_i(x_i).P_i\} \qquad !u(x) \qquad \overline{u}\langle v\rangle$$

$$\uparrow^m (T) = \oplus^m_{i\in\{1\}}\{l_i(T)\} \qquad \downarrow^m (T)$$

## Encoding base types using branching

Base types seen through interaction types:

$$\mathtt{N}^\circ \stackrel{\text{def}}{=} \downarrow^\omega (\oplus^1_{i\in\mathtt{N}}\{\mathbf{i}()\}) \qquad \mathtt{B}^\circ \stackrel{\text{def}}{=} \downarrow^\omega (\oplus^1_{i\in\mathtt{B}}\{\mathbf{i}()\}) \qquad \bigstar^\circ \stackrel{\text{def}}{=} \downarrow^\omega (\oplus^1_{i\in\bigstar}\{\mathbf{i}()\})$$

# Example: a memory cell

$$\mathbf{Mem} = !ref(c, v).c\&^{\overline{1}}_{l \in \{\mathtt{set},\mathtt{get}\}}\left\{ \begin{array}{ll} \mathtt{set}(r, n). & (\overline{r}\langle () \rangle \mid \overline{ref}\langle c, n \rangle) \\ \mathtt{get}(r). & (\overline{r}\langle v \rangle \mid \overline{ref}\langle c, v \rangle) \end{array} \right\}$$

$$\text{in } \mathbf{E}_1 = \ \mathbf{Mem} \mid \overline{ref}\langle cell_1, 0 \rangle \mid \overline{ref}\langle cell_2, 3 \rangle$$

- **Mem** creates memory references.
- two "methods" set and get.
- Type-checked with:

$$T_{ref} = \downarrow^{\omega} (\&^{\overline{1}}_{l \in \{\mathtt{set},\mathtt{get}\}}\left\{ \begin{array}{l} \mathtt{set}(\uparrow^{\overline{1}} (\bigstar), \mathtt{N}) \\ \mathtt{get}(\uparrow^{\overline{1}} (\mathtt{N})) \end{array} \right\}, \mathtt{N})$$

- Concurrence: value of $x$ in

$$\mathbf{E}_1 \mid (\nu ans_1)\ (\overline{c_1} \oplus \mathtt{set}\langle 1, ans_1 \rangle \mid ans_1) \mid (\nu ans_2)\ (\overline{c_1} \oplus \mathtt{get}\langle ans_2 \rangle \mid ans_2(x))$$

# Subtyping

## Definition

$T_1 \sqsubseteq T_2$ when:

- $T_{1,2}$ inputs, and $T_1$ offers more choices than $T_2$.
- $T_{1,2}$ outputs, and $T_1$ selects among less options than $T_2$.

Property: $T_1 \sqsubseteq T_2$ iff $\forall T, T \asymp T_2 \Rightarrow T \asymp T_1$.

# Subtyping

## Definition

$T_1 \sqsubseteq T_2$ when:

- $T_{1,2}$ inputs, and $T_1$ offers more choices than $T_2$.
- $T_{1,2}$ outputs, and $T_1$ selects among less options than $T_2$.

Property: $T_1 \sqsubseteq T_2$ iff $\forall T, T \asymp T_2 \Rightarrow T \asymp T_1$.

## Example

Adding a "delete" method:

$$T'_{ref} = \downarrow^{\omega} \left(\&^{\overline{1}}_{l \in \{\mathtt{set},\mathtt{get},\mathtt{del}\}} \left\{ \begin{array}{l} \mathtt{set}(\uparrow^{\overline{1}}(\bigstar), \mathtt{N}) \\ \mathtt{get}(\uparrow^{\overline{1}}(\mathtt{N})) \\ \mathtt{del}(\uparrow^{\overline{1}}(\bigstar)) \end{array} \right\}, \mathtt{N}\right)$$

We have $T'_{ref} \sqsubseteq T_{ref}$.

$$\mathbf{Mem}' = !ref'(c, v).c\&^{\overline{1}}_{l \in \{\mathtt{set},\mathtt{get},\mathtt{del}\}} \left\{ \begin{array}{ll} \mathtt{set}(r, new). & (\overline{r}\langle() \rangle \mid \overline{ref'}\langle c, new\rangle) \\ \mathtt{get}(r). & (\overline{r}\langle v\rangle \mid \overline{ref'}\langle c, v\rangle) \\ \mathtt{del}(r). & (\overline{r}\langle()\rangle) \end{array} \right\}$$

# Embedding a functional calculus

# The calculus $\lambda^{\Pi,\Sigma,\sqsubseteq}$

$$M ::= M\ M \mid x \mid () \mid \lambda x.M \mid \{l_i.M_i\}_{i\in I} \mid M.l$$

$$\mid\ \ \mathtt{inj}_l(M) \mid \mathtt{case}\ M\ \mathtt{of}\ [l_i(x_i).M_i]_{i\in I} \mid \mathtt{Y}\ M$$

$$T ::= \bigstar \mid \mathtt{N} \mid T \to T \mid \Pi\{l_i : T_i\}_{i\in I} \mid \Sigma[l_i : T_i]_{i\in I}$$

- PCF-like $\lambda$-calculus.
- Records and variants.
- Standard types (products, sums).

$$\frac{}{\lambda x.M\ V \rightarrowtriangle M\{V/x\}} \qquad \frac{j \in I}{\{l_i : M_i\}_{i\in I}.l_j \rightarrowtriangle M_j}$$

$$\frac{j \in I}{\mathtt{case}\ \mathtt{inj}_{l_j}(V)\ \mathtt{of}\ [l_i(x_i).M_i]_{i\in I} \rightarrowtriangle M_j\{V/x_j\}} \qquad \frac{}{\mathtt{Y}\ V \longrightarrow V\ (\mathtt{Y}\ V)}$$

# Types embedding

$$\langle \bigstar \rangle = \uparrow^{\overline{1}} (\bigstar) \qquad \langle \Pi\{l_i : T_i\}_{i \in I} \rangle = \uparrow^{\overline{1}} (\&^{\omega}_{i \in I}\{l_i(\langle T_i \rangle)\})$$

$$\langle \Sigma[l_i : T_i]_{i \in I} \rangle = \uparrow^{\overline{1}} (\oplus^{\omega}_{i \in I}\{l_i(\overline{\langle T_i \rangle})\})$$

$$\langle T_1 \to T_2 \rangle = \uparrow^{\overline{1}} (\&^{m}_{i \in I}\{l_i(\overline{T_i}, \langle T_2 \rangle)\}) \quad \text{if } \langle T_1 \rangle = \oplus^{m}_{i \in I}\{l_i(T_i)\}$$

- Product type $\longrightarrow$ choice type.
- Sum type $\longrightarrow$ selection type.
- Arrow type $\longrightarrow$ choice type:
  - Decomposition of the argument.
  - $\mathtt{B} \to T$ becomes $\uparrow^{\overline{1}} (\&^{\omega}_{l \in \{\mathtt{true},\mathtt{false}\}}\{\mathtt{true}(T); \mathtt{false}(T)\})$

# Terms embedding

$$\begin{array}{ll}
\langle M\ N\rangle_u^\zeta = & (\nu m, n)\ (\langle M\rangle_m^\zeta \mid \langle N\rangle_n^\zeta \mid m(y).n\&_{i\in I}^\omega\{l_i(x_i).y \oplus^\omega l_i\langle x_i, u\rangle\}) \\
& \text{if } M \text{ has type } T \to T' \text{ with } \langle T\rangle = \oplus_{i\in I}^\omega\{l_i(T_i)\} \\
\langle \lambda x.M\rangle_u^\zeta = & (\nu c)\ (\overline{u}\langle c\rangle.c\&_{i\in I}^\omega\{l_i(x_i, m).\langle M\rangle_m^{\zeta, x\mapsto(l_i,x_i)}\}) \\
& \text{if } \lambda x.M \text{ has type } T \to T' \text{ with } \langle T\rangle = \oplus_{i\in I}^\omega\{l_i(T_i)\}
\end{array}$$

$$\langle x\rangle_u^{\zeta.x\mapsto(l_i,x_i)} = \overline{u} \oplus^\omega l_i\langle x_i\rangle$$

- Close to the standard $\lambda \to \pi$ encoding.
- Parametrised by:
  - return channel $u$
  - environment $\zeta$: $\lambda$-variable $x$ to choice $(l_i, x_i)$
- Inside indirections: decomposition of arguments.

# Example: Option type

$$F^{opt} = \lambda x.\mathtt{case}\ x\ \mathtt{of}\ [\ \begin{matrix}\mathtt{some}(x) & (F\ x)\\ \mathtt{none}(x) & 0\end{matrix}\ ]_{l\in\{\mathtt{some},\mathtt{none}\}}$$

- $F^{opt} : (\mathtt{some} : \mathtt{N}) + (\mathtt{none} : ()) \to \mathtt{N}$
  - partial version of the function $F : \mathtt{N} \to \mathtt{N}$
- Encoding in $\pi^{\{1,\overline{1},\omega\}}$:

$$(\nu v)\ \overline{u}\langle v\rangle.v\&^{\overline{1}}_{l\in\{\mathtt{some},\mathtt{none}\}}\{\ \begin{matrix}\mathtt{some}(x,b). & (\nu f,a)\ (\langle F\rangle^{\emptyset}_{f} \mid \overline{a}\oplus^{\overline{1}}\mathtt{some}\langle x\rangle \mid \textbf{App})\\ \mathtt{none}(x,b). & \overline{b}\langle 0\rangle\end{matrix}\ \}$$

where **App** is $f(y).a\&^{\overline{1}}_{l\in\{\mathtt{some},\mathtt{none}\}}\{\ \begin{matrix}\mathtt{some}(z). & \overline{y}\oplus^{\overline{1}}\mathtt{some}\langle z,b\rangle\\ \mathtt{none}(z). & \overline{y}\oplus^{\overline{1}}\mathtt{none}\langle z,b\rangle\end{matrix}\ \}$

- Indirection inside **App**.

# Full abstraction and subtyping

## Theorem

Let $M_1, M_2$ be two $\lambda^{\Pi,\Sigma,\sqsubseteq}$ terms, then $M_1 \simeq_\lambda M_2$ if and only if $\langle M_1 \rangle_u^\emptyset \simeq_\pi \langle M_1 \rangle_u^\emptyset$ under sequential typing.

- $\simeq_\lambda$: Morris's congruence.
- $\simeq_\pi$: barbed congruence.
- Proof technique:
  - Restriction of $\pi^{\{1,\bar{1},\omega\}}$: deterministic $\pi$.
  - Finite definability.

## Subtyping

- Standard subtyping for $\lambda^{\Pi,\Sigma,\sqsubseteq}$ (products, sums).
- Theorem: If $T_1 \sqsubseteq T_2$, then $\langle T_1 \rangle \sqsubseteq \langle T_2 \rangle$.

# Embedding a session-based calculus

# $\pi^{\texttt{session}}$ - a session-based calculus

$$P ::= u(x).P \mid \overline{u}(x).P \mid k!l\langle v\rangle.P \mid k?\{(x_i).P_i\}_{i\in I} \mid \text{if } e \text{ then } P \text{ else } P$$

$$\mid (\nu a)P \mid P|P \mid (\mu X(\widetilde{x}).P)\langle\widetilde{v}\rangle \mid X\langle\widetilde{v}\rangle \mid \mathbf{0}$$

- Two kinds of names $a, b, c$: channels: $u, v$ / sessions: $k, s$.
- Notable features:
  - Synchronous outputs.
  - Session types (with subtyping):
    - Types:

$$T, S ::= \&^{\text{ses}}_{i\in I}\{l_i(T_i).S_i\} \mid \oplus^{\text{ses}}_{i\in I}\{l_i(T_i).S_i\}$$

$$\mid \downarrow^{\text{ses}}(S) \mid \uparrow^{\text{ses}}(S) \mid \mu S.S \mid \mathtt{N} \mid \bigstar$$

    - Output rule:

$$(\textbf{SSel})\frac{\Gamma(v) = T'_j \qquad \Gamma, s : S_j \vdash_{\text{ses}} P \qquad T'_j \sqsubseteq \overline{T_j}}{\Gamma, s : \oplus^{\text{ses}}_{i\in I}\{l_i(\overline{T_i}).S_i\} \vdash_{\text{ses}} s!l_j\langle v\rangle.P}$$

# The Embedding

$$[\![u(x).P]\!] = u(x, c).(\overline{c} \mid [\![P]\!]) \qquad [\![\overline{u}(x).P]\!] = (\nu x, c)\ (\overline{u}\langle x, c\rangle \mid c.[\![P]\!])$$

$$[\![k!l\langle e\rangle.P]\!] = (\nu c)\ (k \oplus^{\overline{1}} l\langle e, c\rangle \mid c(s).[\![P]\!]\{s/k\})$$

$$[\![k?\{l_i(x_i).P_i\}_{i\in I}]\!] = (\boldsymbol{\nu} s)\ k\&^{\overline{1}}_{i\in I}\{l_i(x_i, c).([\![P_i]\!]\{s/k\} \mid \overline{c}\langle s\rangle)\}$$

- New names $c$ for synchronising outputs.
- New names $s$ for carrying *session types* information.
- Encoding of types:

$$[\![\&^{\mathsf{ses}}_{i\in I}\{l_i(T_i).S_i\}]\!] = \&^{\omega}_{i\in I}\{l_i([\![T_i]\!], \uparrow^1 (\overline{[\![S_i]\!]}))\}$$

# Example

$$\mathbf{S} \;=\; \overline{a}(x).x?(z_2).x!\langle z_2\rangle \mid a(y).(y!\langle 0\rangle.y?(z_1))$$

- **S**: two subprocesses, one session $(x/y)$.
- Encoding:

$$\begin{aligned}[\![\mathbf{S}]\!] = \quad & (\nu x, c)(\overline{a}\langle x, c\rangle \mid c.(\nu k_2)\ (x(z_2, c_3).(\overline{c_3}\langle k_2\rangle \mid \overline{k_2}\langle z_2, c_4\rangle \mid c_4))) \\ & \mid a(y, c_0).(\overline{c_0} \mid (\nu c_1)\ (\overline{y}\langle 0, c_1\rangle \mid c_1(k_1).(\nu c_2)\ k_1(z_1, c_2).\overline{c_2}))\end{aligned}$$

- Session $y$ becomes $k_1/k_2$ after one synchronisation.
- $c_1$ ensures asynchronous output on $y$.

# Full abstraction and subtyping

**Theorem**

Let $P, Q$ be two $\pi^{\mathtt{session}}$ processes s.t. $\Gamma \vdash_\pi P$ and $\Gamma \vdash_\pi Q$, then $P \simeq_{\mathtt{test}} Q$ if and only if $[\![P]\!] \simeq_{\mathtt{test}} [\![Q]\!]$.

- $\simeq_{\mathtt{test}}$: testing congruence (may/must congruence).
- Proof technique: Definability + Adequacy.

**Subtyping**

If $S_1 \sqsubseteq S_2$, then $[\![S_1]\!] \sqsubseteq [\![S_2]\!]$

# Future works

- Embeddings:
  - Polymorphism (in $\pi$, System $F$), subtyping.
  - Multiparty session types.
- Logical framework for concurrency:
  - Expressing properties of programs with processes
  - Embedding logics (HML, temporal logics).

The End.