Functions as processes

λ and π

- sequential computation is a particular case of concurrenct computation

- 'higher-order substitution' is not primitive in π

- π-calculus encodings of the λ-calculus illustrate some classical programming idioms in π

- works by: Milner [90], Sangiorgi, Boudol, Laneve

Higher-Order π-calculus

- encoding a 'higher-order' process:

$$\begin{array}{c} \overline{a}\langle R\rangle.\mathbf{0} \mid a(x).(x \mid x) \\ \downarrow \\ R \mid R \end{array}$$

Higher-Order π-calculus

- encoding a 'higher-order' process:

$$\overline{a}\langle R\rangle.\mathbf{0} \mid a(x).(x \mid x) \qquad\qquad (\boldsymbol{\nu} q)\,(\overline{a}\langle q\rangle.q.R) \;\mid\; a(x).(\overline{x}.\mathbf{0} \mid \overline{x}.\mathbf{0})$$

$$\downarrow \qquad\qquad\qquad\qquad\qquad\qquad \downarrow$$

$$R \mid R$$

Higher-Order π-calculus

- encoding a 'higher-order' process:

$$\begin{array}{c} \overline{a}\langle R\rangle.\mathbf{0} \mid a(x).(x \mid x) \\ \downarrow \\ R \mid R \end{array}$$

$$\begin{array}{c} (\boldsymbol{\nu} q)\,(\overline{a}\langle q\rangle.q.R) \;\mid\; a(x).(\overline{x}.\mathbf{0} \mid \overline{x}.\mathbf{0}) \\ \downarrow \\ (\boldsymbol{\nu} q)\,(q.R \mid \overline{q}.\mathbf{0} \mid \overline{q}.\mathbf{0}) \\ \downarrow \\ \downarrow \\ (\boldsymbol{\nu} q)\,(R \mid \mathbf{0} \mid R \mid \mathbf{0}) \quad \sim \quad R \mid R \end{array}$$

Higher-Order π-calculus

- encoding a 'higher-order' process:

$$\begin{array}{c} \overline{a}\langle R\rangle.\mathbf{0} \mid a(x).(x \mid x) \\ \downarrow \\ R \mid R \end{array} \qquad \begin{array}{c} (\boldsymbol{\nu} q)\,(\overline{a}\langle q\rangle.q.R) \;\mid\; a(x).(\overline{x}.\mathbf{0} \mid \overline{x}.\mathbf{0}) \\ \downarrow \\ (\boldsymbol{\nu} q)\,(q.R \mid \overline{q}.\mathbf{0} \mid \overline{q}.\mathbf{0}) \\ \downarrow \\ \downarrow \\ (\boldsymbol{\nu} q)\,(R \mid \mathbf{0} \mid R \mid \mathbf{0}) \quad \sim \quad R \mid R \end{array}$$

- in $HO\pi$, *parametrized processes* (abstractions) are passed
  $\overline{n}\langle (x).P\rangle \mid n(X).\Big(X\lceil a\rceil \mid X\lceil b\rceil\Big) \quad \longrightarrow \quad P_{\{x\leftarrow a\}} \mid P_{\{x\leftarrow b\}}$

Higher-Order π-calculus

- encoding a 'higher-order' process:

$$\overline{a}\langle R\rangle.\mathbf{0} \mid a(x).(x \mid x) \qquad (\boldsymbol{\nu}q)\,(\overline{a}\langle q\rangle.q.R) \mid a(x).(\overline{x}.\mathbf{0} \mid \overline{x}.\mathbf{0})$$

$$\downarrow \qquad\qquad \downarrow$$

$$R \mid R \qquad (\boldsymbol{\nu}q)\,(q.R \mid \overline{q}.\mathbf{0} \mid \overline{q}.\mathbf{0})$$

$$\downarrow$$

$$\downarrow$$

$$(\boldsymbol{\nu}q)\,(R \mid \mathbf{0} \mid R \mid \mathbf{0}) \quad \sim \quad R \mid R$$

- in $HO\pi$, *parametrized processes* (abstractions) are passed

$$\overline{n}\langle (x).P\rangle \mid n(X).\Big(X\lceil a\rceil \mid X\lceil b\rceil\Big) \quad \longrightarrow \quad P_{\{x\leftarrow a\}} \mid P_{\{x\leftarrow b\}}$$

- there exists a *fully abstract* encoding from $HO\pi$ into π

$$P \approx_{HO} Q \qquad \text{iff} \qquad [\![P]\!] \approx [\![Q]\!]$$

(see [SW01])

The λ-calculus

- terms: $M, N \quad ::= \quad \lambda x.\, M \mid x \mid (M\, N)$

The λ-calculus

- terms: $M, N \quad ::= \quad \lambda x.\, M \mid x \mid (M\ N)$
- strategies:

▷ call-by-name

$$(\lambda x.\, M)\, N \longrightarrow M\{N/x\} \qquad \frac{M \longrightarrow M'}{M\ N \longrightarrow M'\ N}$$

The λ-calculus

- terms: $M, N \quad ::= \quad \lambda x.\, M \mid x \mid (M\ N)$

- strategies:

▷ call-by-name

$$(\lambda x.\, M)\, N \longrightarrow M\{N/x\} \qquad \frac{M \longrightarrow M'}{M\ N \longrightarrow M'\ N}$$

▷ call-by-value $\qquad V \ ::= \ \lambda x.\, M \mid x \qquad$ values

The λ-calculus

- terms: $M, N \quad ::= \quad \lambda x.\, M \mid x \mid (M\ N)$

- strategies:

▷ call-by-name

$$(\lambda x.\, M)\, N \;\longrightarrow\; M\{N/x\} \qquad \frac{M \longrightarrow M'}{M\ N \longrightarrow M'\,N}$$

▷ call-by-value $\quad V \;::=\; \lambda x.\, M \mid x \quad$ values

$$(\lambda x.\, M)\, V \;\longrightarrow\; M\{V/x\} \qquad \frac{M \longrightarrow M'}{M\ N \longrightarrow M'\,N} \qquad \frac{N \longrightarrow N'}{V\ N \longrightarrow V\ N'}$$

Encodings: commitments

- forms of interaction: β-reduction vs name passing

  application is rendered using parallel composition

Encodings: commitments

- forms of interaction: β-reduction vs name passing

  application is rendered using parallel composition

  . . . and restriction, to prevent interefernces

Encodings: commitments

- forms of interaction: β-reduction vs name passing

  application is rendered using parallel composition

  ...and restriction, to prevent interefernces

- in the λ-calculus, λ is *the only port*

  in π, the encoding of a term is *located* at some channel

  $\longrightarrow$ a parametrised encoding, $[\![M]\!]_p$

Encodings: commitments

- forms of interaction: β-reduction vs name passing

  application is rendered using parallel composition

  ... and restriction, to prevent interefernces

- in the λ-calculus, λ is *the only port*

  in π, the encoding of a term is *located* at some channel

  $\longrightarrow$ a parametrised encoding, $[\![M]\!]_p$

- strategies (the flow of computation) are encoded using prefixes

Encoding of call-by-name

We start with call-by-name:

- computations in π evolve in an 'outermost' fashion
- only one kind of value: functions

Encoding of call-by-name

We start with call-by-name:

- computations in π evolve in an 'outermost' fashion
- only one kind of value: functions

$$[\![\lambda x.M]\!]_p \stackrel{\text{def}}{=} (\boldsymbol{\nu} v)\, \overline{p}\langle v\rangle.!v(x,q).[\![M]\!]_q$$

(cf. the extra argument in CPS translations)

Encoding of call-by-name

We start with call-by-name:

- computations in π evolve in an 'outermost' fashion
- only one kind of value: functions

$$[\![\lambda x.M]\!]_p \stackrel{\text{def}}{=} (\boldsymbol{\nu} v)\,\overline{p}\langle v\rangle.!v(x,q).[\![M]\!]_q$$

(cf. the extra argument in CPS translations)

N.B.: convention: 'reuse' of variable names

Encoding of call-by-name

We start with call-by-name:

- computations in π evolve in an 'outermost' fashion
- only one kind of value: functions

$$[\![\lambda x.M]\!]_p \stackrel{\text{def}}{=} (\boldsymbol{\nu} v)\, \overline{p}\langle v\rangle.!v(x,q).[\![M]\!]_q$$

(cf. the extra argument in CPS translations)

N.B.: convention: 'reuse' of variable names

$$[\![x]\!]_p \stackrel{\text{def}}{=} \overline{x}\langle p\rangle$$

$$[\![M\ N]\!]_p \stackrel{\text{def}}{=} (\boldsymbol{\nu} q)\left([\![M]\!]_q \mid q(v).(\boldsymbol{\nu} x)\, \overline{v}\langle x, p\rangle.!x(r).[\![N]\!]_r\right)$$

$$\frac{M \longrightarrow M'}{M\ N \longrightarrow M'\ N} \qquad (\lambda x.\, M)\, N \longrightarrow M\{N/x\}$$

An example

$$
\begin{array}{rcl}
[\![\lambda x.M]\!]_p & \stackrel{\mathsf{def}}{=} & (\boldsymbol{\nu} v)\,\overline{p}\langle v\rangle.!v(x,q).[\![M]\!]_q \\
[\![x]\!]_p & \stackrel{\mathsf{def}}{=} & \overline{x}\langle p\rangle \\
[\![M\ N]\!]_p & \stackrel{\mathsf{def}}{=} & (\boldsymbol{\nu} q)\left([\![M]\!]_q \mid q(v).(\boldsymbol{\nu} x)\,\overline{v}\langle x,p\rangle.!x(r).[\![N]\!]_r\right)
\end{array}
$$

$$
[\![I\ y]\!]_p \quad = \quad (\boldsymbol{\nu} q)\left([\![I]\!]_q \mid q(v).(\boldsymbol{\nu} x)\,\overline{v}\langle x,p\rangle.!x(r).[\![y]\!]_r\right)
$$

An example

$$
\begin{array}{rcl}
[\![\lambda x.M]\!]_p & \stackrel{\text{def}}{=} & (\boldsymbol{\nu} v)\,\overline{p}\langle v\rangle.!v(x,q).[\![M]\!]_q \\
[\![x]\!]_p & \stackrel{\text{def}}{=} & \overline{x}\langle p\rangle \\
[\![M\ N]\!]_p & \stackrel{\text{def}}{=} & (\boldsymbol{\nu} q)\left([\![M]\!]_q \mid q(v).(\boldsymbol{\nu} x)\,\overline{v}\langle x,p\rangle.!x(r).[\![N]\!]_r\right)
\end{array}
$$

$$
\begin{array}{rcl}
[\![I\ y]\!]_p & = & (\boldsymbol{\nu} q)\left([\![I]\!]_q \mid q(v).(\boldsymbol{\nu} x)\,\overline{v}\langle x,p\rangle.!x(r).[\![y]\!]_r\right) \\
& = & (\boldsymbol{\nu} q)\Big((\boldsymbol{\nu} v_1)\,(\overline{q}\langle v_1\rangle.!v_1(x_1,q_1).\overline{x_1}\langle q_1\rangle) \\
& & \qquad \mid q(v).(\boldsymbol{\nu} x)\,\overline{v}\langle x,p\rangle.!x(r).\overline{y}\langle r\rangle\Big)
\end{array}
$$

An example

$$
\begin{aligned}
[\![\lambda x.M]\!]_p &\stackrel{\text{def}}{=} (\boldsymbol{\nu} v)\,\overline{p}\langle v\rangle.!v(x,q).[\![M]\!]_q \\
[\![x]\!]_p &\stackrel{\text{def}}{=} \overline{x}\langle p\rangle \\
[\![M\ N]\!]_p &\stackrel{\text{def}}{=} (\boldsymbol{\nu} q)\,\big([\![M]\!]_q \mid q(v).(\boldsymbol{\nu} x)\,\overline{v}\langle x,p\rangle.!x(r).[\![N]\!]_r\big)
\end{aligned}
$$

$$
\begin{aligned}
[\![I\ y]\!]_p &= (\boldsymbol{\nu} q)\,\Big([\![I]\!]_q \mid q(v).(\boldsymbol{\nu} x)\,\overline{v}\langle x,p\rangle.!x(r).[\![y]\!]_r\Big) \\
&= (\boldsymbol{\nu} q)\,\Big((\boldsymbol{\nu} v_1)\,(\overline{q}\langle v_1\rangle.!v_1(x_1,q_1).\overline{x_1}\langle q_1\rangle) \\
&\qquad\qquad \mid q(v).(\boldsymbol{\nu} x)\,\overline{v}\langle x,p\rangle.!x(r).\overline{y}\langle r\rangle\Big) \\
&\longrightarrow (\boldsymbol{\nu} q,v_1)\,\Big(!v_1(x_1,q_1).\overline{x_1}\langle q_1\rangle \mid (\boldsymbol{\nu} x)\,\overline{v_1}\langle x,p\rangle.!x(r).\overline{y}\langle r\rangle\Big)
\end{aligned}
$$

An example

$$
\begin{array}{rcl}
[\![\lambda x.M]\!]_p & \stackrel{\text{def}}{=} & (\boldsymbol{\nu} v)\, \overline{p}\langle v\rangle.!v(x,q).[\![M]\!]_q \\
[\![x]\!]_p & \stackrel{\text{def}}{=} & \overline{x}\langle p\rangle \\
[\![M\ N]\!]_p & \stackrel{\text{def}}{=} & (\boldsymbol{\nu} q)\,\big([\![M]\!]_q \mid q(v).(\boldsymbol{\nu} x)\,\overline{v}\langle x,p\rangle.!x(r).[\![N]\!]_r\big)
\end{array}
$$

$$
\begin{array}{rcl}
[\![I\ y]\!]_p & = & (\boldsymbol{\nu} q)\Big([\![I]\!]_q \mid q(v).(\boldsymbol{\nu} x)\,\overline{v}\langle x,p\rangle.!x(r).[\![y]\!]_r\Big) \\
& = & (\boldsymbol{\nu} q)\Big((\boldsymbol{\nu} v_1)\,(\overline{q}\langle v_1\rangle.!v_1(x_1,q_1).\overline{x_1}\langle q_1\rangle) \\
& & \qquad\qquad \mid q(v).(\boldsymbol{\nu} x)\,\overline{v}\langle x,p\rangle.!x(r).\overline{y}\langle r\rangle\Big) \\
& \longrightarrow & (\boldsymbol{\nu} q, v_1)\Big(!v_1(x_1,q_1).\overline{x_1}\langle q_1\rangle \mid (\boldsymbol{\nu} x)\,\overline{v_1}\langle x,p\rangle.!x(r).\overline{y}\langle r\rangle\Big) \\
& \longrightarrow & (\boldsymbol{\nu} x)\Big(\overline{x}\langle p\rangle \mid !x(r).\overline{y}\langle r\rangle\Big)
\end{array}
$$

An example

$$
\begin{array}{rcl}
[\![\lambda x.M]\!]_p & \stackrel{\text{def}}{=} & (\boldsymbol{\nu} v)\, \overline{p}\langle v\rangle.!v(x,q).[\![M]\!]_q \\
[\![x]\!]_p & \stackrel{\text{def}}{=} & \overline{x}\langle p\rangle \\
[\![M\ N]\!]_p & \stackrel{\text{def}}{=} & (\boldsymbol{\nu} q)\,\big([\![M]\!]_q \mid q(v).(\boldsymbol{\nu} x)\,\overline{v}\langle x,p\rangle.!x(r).[\![N]\!]_r\big)
\end{array}
$$

$$
\begin{array}{rcl}
[\![I\ y]\!]_p & = & (\boldsymbol{\nu} q)\Big([\![I]\!]_q \mid q(v).(\boldsymbol{\nu} x)\,\overline{v}\langle x,p\rangle.!x(r).[\![y]\!]_r\Big) \\
& = & (\boldsymbol{\nu} q)\Big((\boldsymbol{\nu} v_1)\,(\overline{q}\langle v_1\rangle.!v_1(x_1,q_1).\overline{x_1}\langle q_1\rangle) \\
& & \qquad\qquad \mid q(v).(\boldsymbol{\nu} x)\,\overline{v}\langle x,p\rangle.!x(r).\overline{y}\langle r\rangle\Big) \\
& \longrightarrow & (\boldsymbol{\nu} q, v_1)\Big(!v_1(x_1,q_1).\overline{x_1}\langle q_1\rangle \mid (\boldsymbol{\nu} x)\,\overline{v_1}\langle x,p\rangle.!x(r).\overline{y}\langle r\rangle\Big) \\
& \longrightarrow & (\boldsymbol{\nu} x)\Big(\overline{x}\langle p\rangle \mid !x(r).\overline{y}\langle r\rangle\Big) \\
& \longrightarrow & (\boldsymbol{\nu} x)\Big(!x(r).\overline{y}\langle r\rangle\Big) \quad \mid \quad \overline{y}\langle p\rangle
\end{array}
$$

An example

$$
\begin{aligned}
[\![\lambda x.M]\!]_p &\stackrel{\text{def}}{=} (\boldsymbol{\nu} v)\, \overline{p}\langle v\rangle.!v(x,q).[\![M]\!]_q \\
[\![x]\!]_p &\stackrel{\text{def}}{=} \overline{x}\langle p\rangle \\
[\![M\; N]\!]_p &\stackrel{\text{def}}{=} (\boldsymbol{\nu} q)\, \big([\![M]\!]_q \mid q(v).(\boldsymbol{\nu} x)\, \overline{v}\langle x,p\rangle.!x(r).[\![N]\!]_r\big)
\end{aligned}
$$

$$
\begin{aligned}
[\![I\; y]\!]_p &= (\boldsymbol{\nu} q)\, \Big([\![I]\!]_q \mid q(v).(\boldsymbol{\nu} x)\, \overline{v}\langle x,p\rangle.!x(r).[\![y]\!]_r\Big) \\
&= (\boldsymbol{\nu} q)\, \Big((\boldsymbol{\nu} v_1)\, (\overline{q}\langle v_1\rangle.!v_1(x_1,q_1).\overline{x_1}\langle q_1\rangle) \\
&\qquad\qquad \mid q(v).(\boldsymbol{\nu} x)\, \overline{v}\langle x,p\rangle.!x(r).\overline{y}\langle r\rangle\Big) \\
&\longrightarrow (\boldsymbol{\nu} q, v_1)\, \Big(!v_1(x_1,q_1).\overline{x_1}\langle q_1\rangle \mid (\boldsymbol{\nu} x)\, \overline{v_1}\langle x,p\rangle.!x(r).\overline{y}\langle r\rangle\Big) \\
&\longrightarrow (\boldsymbol{\nu} x)\, \Big(\overline{x}\langle p\rangle \mid !x(r).\overline{y}\langle r\rangle\Big) \\
&\longrightarrow (\boldsymbol{\nu} x)\, \Big(!x(r).\overline{y}\langle r\rangle\Big) \mid \overline{y}\langle p\rangle \quad \sim \quad \overline{y}\langle p\rangle
\end{aligned}
$$

Encoding of call-by-value

$$
\begin{aligned}
[\![\lambda x.M]\!]_p &\stackrel{\text{def}}{=} (\boldsymbol{\nu} v)\,\overline{p}\langle v\rangle.!v(x,q).[\![M]\!]_q \\
[\![x]\!]_p &\stackrel{\text{def}}{=} \overline{x}\langle p\rangle \\
[\![M\;N]\!]_p &\stackrel{\text{def}}{=} (\boldsymbol{\nu} q)\Big([\![M]\!]_q \\
&\qquad \mid q(v).(\boldsymbol{\nu} r)\Big([\![N]\!]_r \mid r(w).(\boldsymbol{\nu} x)\,\overline{v}\langle x,p\rangle.!x(r').\overline{r'}\langle w\rangle\Big)\Big)
\end{aligned}
$$

$$
\frac{M \longrightarrow M'}{M\;N \longrightarrow M'\;N} \qquad \frac{N \longrightarrow N'}{V\;N \longrightarrow V\;N'} \qquad (\lambda x.\,M)\,V \longrightarrow M\{V/x\}
$$

Exercise: the encodings revisited

- call-by-name: the encoding we have given can be optimised; how?

Exercise: the encodings revisited

- call-by-name: the encoding we have given can be optimised; how?

- call-by-value: many indirections
  $\longrightarrow$ write a more efficient encoding where $[\![x]\!]_p \stackrel{\text{def}}{=} \overline{p}\langle x\rangle$

Exercises: enriched λ-calculi

- how should we adapt the encoding for a λ-calculus enriched with a *parallel convergence test* operator $\mathtt{P}$?

$$\frac{M \downarrow}{\mathtt{P}\, M\, N \;\longrightarrow\; I} \qquad \frac{M \downarrow}{\mathtt{P}\, N\, M \;\longrightarrow\; I}$$

$$\frac{M \;\longrightarrow\; M'}{\mathtt{P}\, M\, N \;\longrightarrow\; \mathtt{P}\, M'\, N} \quad \frac{M \;\longrightarrow\; M'}{\mathtt{P}\, N\, M \;\longrightarrow\; \mathtt{P}\, N\, M'}$$

Exercises: enriched λ-calculi

- how should we adapt the encoding for a λ-calculus enriched with a *parallel convergence test* operator $\mathtt{P}$?

$$\frac{M \downarrow}{\mathtt{P}\, M\, N \;\longrightarrow\; I} \qquad \frac{M \downarrow}{\mathtt{P}\, N\, M \;\longrightarrow\; I}$$

$$\frac{M \;\longrightarrow\; M'}{\mathtt{P}\, M\, N \;\longrightarrow\; \mathtt{P}\, M'\, N} \qquad \frac{M \;\longrightarrow\; M'}{\mathtt{P}\, N\, M \;\longrightarrow\; \mathtt{P}\, N\, M'}$$

- idem for the encoding of the call-by-name λ-calculus enriched with a `let...in` construct to perform *local call-by-value*

Types of the names used in the encoding

- call-by-name encoding, again:

$$
\begin{array}{rcl}
[\![\lambda x.M]\!]_p & \stackrel{\text{def}}{=} & (\boldsymbol{\nu} v \qquad)\, \overline{p}\langle v\rangle.!v(x,q).[\![M]\!]_q \\
[\![x]\!]_p & \stackrel{\text{def}}{=} & \overline{x}\langle p\rangle \\
[\![M\ N]\!]_p & \stackrel{\text{def}}{=} & (\boldsymbol{\nu} q \qquad)\, \big([\![M]\!]_q \mid q(v).(\boldsymbol{\nu} x \qquad)\, \overline{v}\langle x,p\rangle.!x(r).[\![N]\!]_r\big)
\end{array}
$$

define $\mathsf{Val} \stackrel{\text{def}}{=} \mu X.\, \mathsf{o}\langle \mathsf{oo}X, \mathsf{o}X\rangle$ and $\mathsf{Val}^{\updownarrow} \stackrel{\text{def}}{=} \#\langle \mathsf{ooVal}, \mathsf{oVal}\rangle$

$\mathsf{Val}^{\updownarrow}$: *(unfolding Val and replacing the outermost* o *with a* $\#$*)*

Types of the names used in the encoding

- call-by-name encoding, again:

$$
\begin{aligned}
[\![\lambda x.M]\!]_p &\stackrel{\mathsf{def}}{=} (\boldsymbol{\nu} v \colon \mathsf{Val}^{\updownarrow})\, \overline{p}\langle v\rangle.!v(x,q).[\![M]\!]_q \\
[\![x]\!]_p &\stackrel{\mathsf{def}}{=} \overline{x}\langle p\rangle \\
[\![M\ N]\!]_p &\stackrel{\mathsf{def}}{=} (\boldsymbol{\nu} q \qquad)\big([\![M]\!]_q \mid q(v).(\boldsymbol{\nu} x \qquad)\,\overline{v}\langle x,p\rangle.!x(r).[\![N]\!]_r\big)
\end{aligned}
$$

define $\mathsf{Val} \stackrel{\mathsf{def}}{=} \mu X.\, \mathsf{o}\langle \mathsf{oo}X, \mathsf{o}X\rangle$ and $\mathsf{Val}^{\updownarrow} \stackrel{\mathsf{def}}{=} \#\langle \mathsf{oo}\mathsf{Val}, \mathsf{o}\mathsf{Val}\rangle$

$\mathsf{Val}^{\updownarrow}$: *(unfolding Val and replacing the outermost* o *with a* $\#$*)*

Types of the names used in the encoding

- call-by-name encoding, again:

$$
\begin{aligned}
[\![\lambda x.M]\!]_p &\stackrel{\text{def}}{=} (\boldsymbol{\nu} v{:}\ \mathsf{Val}^{\updownarrow})\, \overline{p}\langle v\rangle.!v(x,q).[\![M]\!]_q \\
[\![x]\!]_p &\stackrel{\text{def}}{=} \overline{x}\langle p\rangle \\
[\![M\,N]\!]_p &\stackrel{\text{def}}{=} (\boldsymbol{\nu} q{:}\ \#\mathsf{Val})\,\big([\![M]\!]_q \mid q(v).(\boldsymbol{\nu} x \qquad)\,\overline{v}\langle x,p\rangle.!x(r).[\![N]\!]_r\big)
\end{aligned}
$$

define $\mathsf{Val} \stackrel{\text{def}}{=} \mu X.\, \mathsf{o}\langle \mathsf{oo}X, \mathsf{o}X\rangle$ and $\mathsf{Val}^{\updownarrow} \stackrel{\text{def}}{=} \#\langle \mathsf{ooVal}, \mathsf{oVal}\rangle$

$\mathsf{Val}^{\updownarrow}$: *(unfolding Val and replacing the outermost* o *with a #)*

Types of the names used in the encoding

- call-by-name encoding, again:

$$\begin{array}{rcl}
[\![\lambda x.M]\!]_p & \stackrel{\mathsf{def}}{=} & (\boldsymbol{\nu} v\colon \mathsf{Val}^{\updownarrow})\, \overline{p}\langle v\rangle.!v(x,q).[\![M]\!]_q \\
[\![x]\!]_p & \stackrel{\mathsf{def}}{=} & \overline{x}\langle p\rangle \\
[\![M\ N]\!]_p & \stackrel{\mathsf{def}}{=} & (\boldsymbol{\nu} q\colon \#\mathsf{Val})\,\big([\![M]\!]_q \mid q(v).(\boldsymbol{\nu} x\colon \#\mathsf{oVal})\,\overline{v}\langle x,p\rangle.!x(r).[\![N]\!]_r\big)
\end{array}$$

define $\mathsf{Val} \stackrel{\mathsf{def}}{=} \mu X.\, \mathsf{o}\langle \mathsf{oo}X, \mathsf{o}X\rangle$ and $\mathsf{Val}^{\updownarrow} \stackrel{\mathsf{def}}{=} \#\langle \mathsf{ooVal}, \mathsf{oVal}\rangle$

$\mathsf{Val}^{\updownarrow}$: *(unfolding Val and replacing the outermost* o *with a* $\#$*)*

Types of the names used in the encoding

- call-by-name encoding, again:

$$
\begin{aligned}
[\![\lambda x.M]\!]_p &\stackrel{\text{def}}{=} (\boldsymbol{\nu} v \colon \mathsf{Val}^{\updownarrow})\, \overline{p}\langle v\rangle.!v(x,q).[\![M]\!]_q \\
[\![x]\!]_p &\stackrel{\text{def}}{=} \overline{x}\langle p\rangle \\
[\![M\ N]\!]_p &\stackrel{\text{def}}{=} (\boldsymbol{\nu} q \colon \#\mathsf{Val})\, \big([\![M]\!]_q \mid q(v).(\boldsymbol{\nu} x \colon \#\mathsf{oVal})\, \overline{v}\langle x,p\rangle.!x(r).[\![N]\!]_r\big)
\end{aligned}
$$

define $\mathsf{Val} \stackrel{\text{def}}{=} \mu X.\, \mathsf{o}\langle \mathsf{oo}X, \mathsf{o}X\rangle$ and $\mathsf{Val}^{\updownarrow} \stackrel{\text{def}}{=} \#\langle \mathsf{ooVal}, \mathsf{oVal}\rangle$

$\mathsf{Val}^{\updownarrow}$: *(unfolding Val and replacing the outermost* o *with a #)*

- typing open terms:
  if $\mathrm{fv}(M) \subseteq \tilde{x}$, then $\tilde{x} : \mathsf{ooVal}, p : \mathsf{oVal} \vdash [\![M]\!]_p$

Correspondence between λ-terms and their encodings

- adequacy: $M \Downarrow$ iff $[\![M]\!]_p \downarrow_\eta$ for some $\eta = a$ or $\overline{a}$

  $M \Downarrow$: M *converges to a value*

Correspondence between λ-terms and their encodings

- adequacy: $M \Downarrow$ iff $[\![M]\!]_p \downarrow_\eta$ for some $\eta = a$ or $\overline{a}$

$M \Downarrow$: M converges to a value

- validity of $\lambda\beta$ theory on closed terms

suppose $\mathrm{fv}(M, N) = \emptyset$ and $\lambda\beta \vdash M = N$, then

$$p : \mathsf{oVal} \triangleright [\![M]\!]_p \cong^c [\![N]\!]_p$$

$\Gamma \triangleright P \cong^c Q$: typed *behavioural equivalence*

this result can be adapted to terms with free variables

Encoding of a typed λ-calculus

- remark: any term of the untyped λ-calculus has type $\mu X.\, X \to X$, which is translated as the type $\mathsf{Val} \; = \; \mu X.\, \mathsf{o}\langle \mathsf{oo}X, \mathsf{o}X \rangle$

Encoding of a typed λ-calculus

- remark: any term of the untyped λ-calculus has type $\mu X.\, X \to X$, which is translated as the type $\mathsf{Val} \;=\; \mu X.\, \mathsf{o}\langle \mathsf{oo}X, \mathsf{o}X\rangle$

- outline of the encoding of types:

  if $\lambda x.\, M : T \to U$, then p has type $\mathsf{oo}\langle [\![T]\!], \mathsf{o}[\![U]\!]\rangle$ in $[\![M]\!]_p$

  (p points to the 'value' $\lambda x.\, M$)

  and $[\![T \to U]\!] \stackrel{\text{def}}{=} \mathsf{o}\langle [\![T]\!], \mathsf{o}[\![U]\!]\rangle$

Encoding of a typed λ-calculus

- remark: any term of the untyped λ-calculus has type $\mu X.\, X \to X$, which is translated as the type $\mathsf{Val} \;=\; \mu X.\, \mathsf{o}\langle \mathsf{oo}X, \mathsf{o}X\rangle$

- outline of the encoding of types:
  if $\lambda x.\, M : T \to U$, then p has type $\mathsf{oo}\langle [\![T]\!], \mathsf{o}[\![U]\!]\rangle$ in $[\![M]\!]_p$
  (p points to the 'value' $\lambda x.\, M$)
  and $[\![T \to U]\!] \stackrel{\text{def}}{=} \mathsf{o}\langle [\![T]\!], \mathsf{o}[\![U]\!]\rangle$

- subtyping with arrow types: $\dfrac{T_1 \leq U_1 \quad T_2 \leq U_2}{U_1 \to T_2 \;\leq\; T_1 \to U_2}$

Encoding of a typed λ-calculus

- remark: any term of the untyped λ-calculus has type $\mu X.\, X \to X$, which is translated as the type $\mathsf{Val} \;=\; \mu X.\, \mathsf{o}\langle \mathsf{oo}X, \mathsf{o}X\rangle$

- outline of the encoding of types:
  if $\lambda x.\, M : T \to U$, then p has type $\mathsf{oo}\langle [\![T]\!], \mathsf{o}[\![U]\!]\rangle$ in $[\![M]\!]_p$
  (p points to the 'value' $\lambda x.\, M$)
  and $[\![T \to U]\!] \overset{\text{def}}{=} \mathsf{o}\langle [\![T]\!], \mathsf{o}[\![U]\!]\rangle$

- subtyping with arrow types: $\dfrac{T_1 \le U_1 \quad T_2 \le U_2}{U_1 \to T_2 \;\le\; T_1 \to U_2}$
  $\longrightarrow$ *odd* number of o : contravariance, *even*: covariance