Mobile Ambients

Mobility

- name passing in the π-calculus models dynamically evolving network topology

Mobility

- name passing in the π-calculus models dynamically evolving network topology

- actually this is enough to represent mobility of terms *(Higher-Order π-calculus, HOπ)*

Mobility

- name passing in the π-calculus models dynamically evolving network topology

- actually this is enough to represent mobility of terms *(Higher-Order π-calculus, HOπ)*

- Mobile Ambients: a model where term mobility is a primitive

From π-calculus to Mobile Ambients

$$\boxed{\pi} \quad P \;=\; \mathbf{0} \mid a(x).P \mid \overline{a}\langle x\rangle.P \mid (P_1|P_2) \mid\ !P \mid (\nu a)\,P \mid P + Q$$

From π-calculus to Mobile Ambients

$$
\begin{array}{lrcl}
\boxed{\pi} & P & = & \mathbf{0} \mid a(x).P \mid \overline{a}\langle x\rangle.P \mid (P_1|P_2) \mid !P \mid (\nu a)\,P \mid P+Q \\
& \downarrow & & \\
& P & = & \mathbf{0} \qquad\qquad\qquad\qquad\qquad\quad \mid (P_1|P_2) \mid !P \mid (\nu a)\,P
\end{array}
$$

constructs for *space*

From π-calculus to Mobile Ambients

$$\boxed{\pi} \quad P \;=\; \mathbf{0} \mid a(x).P \mid \overline{a}\langle x\rangle.P \mid (P_1|P_2) \mid\, !P \mid (\nu a)\,P \mid P+Q$$

$$\downarrow$$

$$P \;=\; \mathbf{0} \qquad\qquad \mid (P_1|P_2) \mid\, !P \mid (\nu a)\,P$$

constructs for *space*

$+$ *localities (ambients)*: $\boxed{n[P]}$

spatial structure

From π-calculus to Mobile Ambients

$$\boxed{\pi} \quad P = \mathbf{0} \mid a(x).P \mid \overline{a}\langle x\rangle.P \mid (P_1|P_2) \mid !P \mid (\nu a)\,P \mid P+Q$$

$$\downarrow$$

$$P = \mathbf{0} \qquad\qquad \mid (P_1|P_2) \mid !P \mid (\nu a)\,P$$

constructs for *space*

+ *localities (ambients)*: $\boxed{n[P]}$

  spatial structure

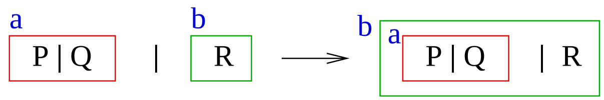
+ actions: *capabilities* $\boxed{M.P}$

  make the spatial structure evolve

Mobile Ambients – [Cardelli Gordon 1998]

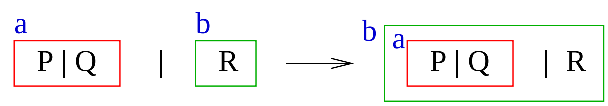
$$
\begin{array}{rcl}
P & == & \mathbf{0} \mid a[P] \mid P_1|P_2 \mid !P \mid (\nu x)\,P \mid M.P \\
M & == & \mathsf{in}\,a \mid \mathsf{out}\,a \mid \mathsf{open}\,a
\end{array}
$$

Reduction, movement

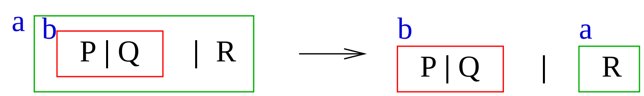


$$a[\mathsf{in}\, b.P \mid Q] \mid b[R] \quad \longrightarrow \quad b[a[P|Q] \mid R]$$

Reduction, movement

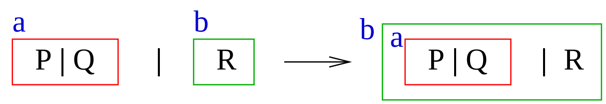


$$a[\mathsf{in}\, b.P \mid Q] \mid b[R] \quad \longrightarrow \quad b[a[P|Q] \mid R]$$

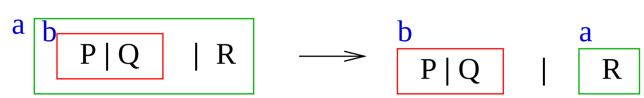


$$a[b[\mathsf{out}\, a.P \mid Q] \mid R] \quad \longrightarrow \quad b[P|Q] \mid a[R]$$

Reduction, movement



$$a[\mathsf{in}\, b.P \mid Q] \mid b[R] \quad\longrightarrow\quad b[a[P|Q] \mid R]$$



$$a[b[\mathsf{out}\, a.P \mid Q] \mid R] \quad\longrightarrow\quad b[P|Q] \mid a[R]$$

Remark: movements are *subjective*

Reduction, opening

$$a\,\boxed{P} \mid Q \longrightarrow P \mid Q$$

$$a[P] \mid \mathsf{open}\, a.Q \quad \rightarrow \quad P|Q$$

Reduction, opening

$$^{a}\boxed{P} \mid Q \longrightarrow P \mid Q$$

$$a[P] \mid \mathsf{open}\, a.Q \quad \to \quad P|Q$$

- opening is “objective”

Reduction, opening

$$^{a}\boxed{P} \mid Q \longrightarrow P \mid Q$$

$$a[P] \mid \mathsf{open}\, a.Q \quad \longrightarrow \quad P|Q$$

- opening is "objective"
- an ambient disappears, its content is thrown in its enclosing ambient (capabilities act on a different locality)

Where does non-determinism come from in Mobile Ambients?

$$
\begin{array}{rcll}
a[\mathsf{in}\, b.P \mid Q] \mid b[R] & \longrightarrow & b[a[P|Q] \mid R] & \text{in} \\
a[b[\mathsf{out}\, a.P \mid Q] \mid R] & \longrightarrow & b[P|Q] \mid a[R] & \text{out} \\
a[P] \mid \mathsf{open}\, a.Q & \longrightarrow & P|Q & \text{open}
\end{array}
$$

Where does non-determinism come from in Mobile Ambients?

$$
\boxed{
\begin{array}{rcll}
a[\mathsf{in}\, b.P \mid Q] \mid b[R] & \longrightarrow & b[a[P|Q] \mid R] & \text{in} \\
a[b[\mathsf{out}\, a.P \mid Q] \mid R] & \longrightarrow & b[P|Q] \mid a[R] & \text{out} \\
a[P] \mid \mathsf{open}\, a.Q & \longrightarrow & P|Q & \text{open}
\end{array}
}
$$

$$
a[P] \mid b[\mathsf{in}\, a.Q \mid Q'] \mid a[R]
$$

Where does non-determinism come from in Mobile Ambients?

$$
\boxed{\begin{array}{rcll}
a[\mathsf{in}\, b.P \mid Q] \mid b[R] & \rightarrow & b[a[P|Q] \mid R] & \quad \text{in} \\
a[b[\mathsf{out}\, a.P \mid Q] \mid R] & \rightarrow & b[P|Q] \mid a[R] & \quad \text{out} \\
a[P] \mid \mathsf{open}\, a.Q & \rightarrow & P|Q & \quad \text{open}
\end{array}}
$$

$$a[P] \mid b[\mathsf{in}\, a.Q \mid Q'] \mid a[R]$$

$$a[b[\mathsf{in}\, b'.P \mid \mathsf{out}\, a.P'] \mid b'[Q]]$$

Is name extrusion available in Mobile Ambients?

- structural congruence rules:

▷ $(\nu n)\,(P \mid Q) \;\equiv\; P \mid (\nu n)\,Q$ if $n \notin \mathsf{fn}(P)$

Is name extrusion available in Mobile Ambients?

- structural congruence rules:

▷ $(\nu n)\,(P \mid Q) \;\equiv\; P \mid (\nu n)\,Q$ if $n \notin \mathsf{fn}(P)$

▷ $(\nu n)\,m[P] \;\equiv\; m[(\nu n)\,P]$ if $n \neq m$

Is name extrusion available in Mobile Ambients?

- structural congruence rules:
  - ▷ $(\nu n)\,(P \mid Q) \;\equiv\; P \mid (\nu n)\,Q$ if $n \notin \mathsf{fn}(P)$
  - ▷ $(\nu n)\,m[P] \;\equiv\; m[(\nu n)\,P]$ if $n \neq m$
- take for example $P \stackrel{\mathsf{def}}{=} a[(\nu c)\,(b[\mathsf{out}\,a.\mathsf{in}\,r.P_c] \mid c[Q])] \mid r[T]$

Is name extrusion available in Mobile Ambients?

- structural congruence rules:
  - ▷ $(\nu n)\,(P \mid Q) \;\equiv\; P \mid (\nu n)\,Q$ if $n \notin \mathsf{fn}(P)$
  - ▷ $(\nu n)\,m[P] \;\equiv\; m[(\nu n)\,P]$ if $n \neq m$
- take for example $P \stackrel{\mathsf{def}}{=} a[(\nu c)\,(b[\mathsf{out}\,a.\mathsf{in}\,r.P_c] \mid c[Q])] \mid r[T]$

$$P \;\rightarrow\; (\nu c)\left(a[c[Q]] \mid b[\mathsf{in}\,r.P_c]\right) \mid r[T]$$

Is name extrusion available in Mobile Ambients?

- structural congruence rules:

▷ $(\nu n)\,(P \mid Q) \;\equiv\; P \mid (\nu n)\,Q$ if $n \notin \mathsf{fn}(P)$

▷ $(\nu n)\,m[P] \;\equiv\; m[(\nu n)\,P]$ if $n \neq m$

- take for example $P \stackrel{\mathsf{def}}{=} a[(\nu c)\,(b[\mathsf{out}\,a.\mathsf{in}\,r.P_c] \mid c[Q])] \mid r[T]$

$$\begin{aligned} P \;&\rightarrow\; (\nu c)\left(a[c[Q]] \mid b[\mathsf{in}\,r.P_c]\right) \mid r[T] \\ &\rightarrow\; (\nu c)\left(a[c[Q]] \mid r[b[P_c] \mid T]\right) \end{aligned}$$

Is name extrusion available in Mobile Ambients?

- structural congruence rules:

▷ $(\nu n)\,(P \mid Q) \;\equiv\; P \mid (\nu n)\,Q$ if $n \notin \mathsf{fn}(P)$

▷ $(\nu n)\,m[P] \;\equiv\; m[(\nu n)\,P]$ if $n \neq m$

- take for example $P \stackrel{\mathsf{def}}{=} a[(\nu c)\,(b[\mathsf{out}\,a.\mathsf{in}\,r.P_c] \mid c[Q])] \mid r[T]$

$$\begin{aligned} P &\;\to\; (\nu c)\left(a[c[Q]] \mid b[\mathsf{in}\,r.P_c]\right) \mid r[T] \\ &\;\to\; (\nu c)\left(a[c[Q]] \mid r[b[P_c] \mid T]\right) \end{aligned}$$

- c is a secret that is carried along

Firewall

- $n[P]$ a firewall named n enclosing P

- capabilities $\mathsf{in}\,n$, $\mathsf{out}\,n$ are needed in order to cross the firewall

- the context $=$ the network

Firewall

- $n[P]$ a firewall named n enclosing P
- capabilities $\mathsf{in}\, n$, $\mathsf{out}\, n$ are needed in order to cross the firewall
- the context = the network

Example: an agent $Agent \;=\; k'[\mathsf{open}\, k.k''[B]]$,
the firewall $Firewall \;=\; w[k[\mathsf{out}\, w.\mathsf{in}\, k'.\mathsf{in}\, w.\mathbf{0}] | \mathsf{open}\, k'.\mathsf{open}\, k''.A]$

$$(\nu w)\,(Agent \mid Firewall) \;\approx\; (\nu w)\, w[B \mid A]$$

$$(\nu w)(w[k[\underline{\mathsf{out}\ w}.\mathsf{in}\ k'.\mathsf{in}\ w.\mathbf{0}] | \mathsf{open}\ k'.\mathsf{open}\ k''.A]) \quad | \quad k'[\mathsf{open}\ k.k''[B]]$$

$$(\nu w)(w[k[\underline{\mathsf{out}\ w}.\mathsf{in}\ k'.\mathsf{in}\ w.\mathbf{0}] | \mathsf{open}\ k'.\mathsf{open}\ k''.A]) \quad | \quad k'[\mathsf{open}\ k.k''[B]]$$

$$\downarrow$$

$$(\nu w)(w[\mathsf{open}\ k'.\mathsf{open}\ k''.A] \quad | \quad k[\underline{\mathsf{in}\ k'}.\mathsf{in}\ w.\mathbf{0}]) \quad | \quad k'[\mathsf{open}\ k.k''[B]]$$

$$(\nu w)(w[k[\underline{\mathsf{out}\ w}.\mathsf{in}\ k'.\mathsf{in}\ w.\mathbf{0}] | \mathsf{open}\ k'.\mathsf{open}\ k''.A]) \quad | \quad k'[\mathsf{open}\ k.k''[B]]$$
$$\downarrow$$
$$(\nu w)(w[\mathsf{open}\ k'.\mathsf{open}\ k''.A] \quad | \quad k[\underline{\mathsf{in}\ k'}.\mathsf{in}\ w.\mathbf{0}]) \quad | \quad k'[\mathsf{open}\ k.k''[B]]$$
$$\downarrow$$
$$(\nu w)(w[\mathsf{open}\ k'.\mathsf{open}\ k''.A] \quad | \quad k'[k[\mathsf{in}\ w.\mathbf{0}] | \underline{\mathsf{open}\ k}.k''[B]])$$

$$(\nu w)(w[k[\underline{\mathsf{out}\ w}.\mathsf{in}\ k'.\mathsf{in}\ w.\mathbf{0}] | \mathsf{open}\ k'.\mathsf{open}\ k''.A]) \quad | \quad k'[\mathsf{open}\ k.k''[B]]$$

$$\downarrow$$

$$(\nu w)(w[\mathsf{open}\ k'.\mathsf{open}\ k''.A] \quad | \quad k[\underline{\mathsf{in}\ k'}.\mathsf{in}\ w.\mathbf{0}]) \quad | \quad k'[\mathsf{open}\ k.k''[B]]$$

$$\downarrow$$

$$(\nu w)(w[\mathsf{open}\ k'.\mathsf{open}\ k''.A] \quad | \quad k'[k[\mathsf{in}\ w.\mathbf{0}] | \underline{\mathsf{open}\ k}.k''[B]])$$

$$\downarrow$$

$$(\nu w)(w[\mathsf{open}\ k'.\mathsf{open}\ k''.A] \quad | \quad k'[\underline{\mathsf{in}\ w}.\mathbf{0} | k''[B]])$$

$$
\begin{array}{c}
(\nu w)(w[k[\underline{\mathsf{out}\,w}.\mathsf{in}\,k'.\mathsf{in}\,w.\mathbf{0}]|\mathsf{open}\,k'.\mathsf{open}\,k''.A]) \quad | \quad k'[\mathsf{open}\,k.k''[B]] \\
\downarrow \\
(\nu w)(w[\mathsf{open}\,k'.\mathsf{open}\,k''.A] \quad | \quad k[\underline{\mathsf{in}\,k'}.\mathsf{in}\,w.\mathbf{0}]) \quad | \quad k'[\mathsf{open}\,k.k''[B]] \\
\downarrow \\
(\nu w)(w[\mathsf{open}\,k'.\mathsf{open}\,k''.A] \quad | \quad k'[k[\mathsf{in}\,w.\mathbf{0}]|\underline{\mathsf{open}\,k}.k''[B]]) \\
\downarrow \\
(\nu w)(w[\mathsf{open}\,k'.\mathsf{open}\,k''.A] \quad | \quad k'[\underline{\mathsf{in}\,w}.\mathbf{0}|k''[B]]) \\
\downarrow \\
(\nu w)w[\underline{\mathsf{open}\,k'}.\mathsf{open}\,k''.A \mid k'[\mathbf{0} \mid k''[B]]]
\end{array}
$$

$$(\nu w)(w[k[\underline{\mathsf{out}\ w}.\mathsf{in}\ k'.\mathsf{in}\ w.\mathbf{0}] | \mathsf{open}\ k'.\mathsf{open}\ k''.A]) \quad | \quad k'[\mathsf{open}\ k.k''[B]]$$

$$\downarrow$$

$$(\nu w)(w[\mathsf{open}\ k'.\mathsf{open}\ k''.A] \quad | \quad k[\underline{\mathsf{in}\ k'}.\mathsf{in}\ w.\mathbf{0}]) \quad | \quad k'[\mathsf{open}\ k.k''[B]]$$

$$\downarrow$$

$$(\nu w)(w[\mathsf{open}\ k'.\mathsf{open}\ k''.A] \quad | \quad k'[k[\mathsf{in}\ w.\mathbf{0}] | \underline{\mathsf{open}\ k}.k''[B]])$$

$$\downarrow$$

$$(\nu w)(w[\mathsf{open}\ k'.\mathsf{open}\ k''.A] \quad | \quad k'[\underline{\mathsf{in}\ w}.\mathbf{0} | k''[B]])$$

$$\downarrow$$

$$(\nu w)w[\underline{\mathsf{open}\ k'}.\mathsf{open}\ k''.A \ | \ k'[\mathbf{0} \ | \ k''[B]]]$$

$$\downarrow$$

$$(\nu w)w[\underline{\mathsf{open}\ k''}.A \ | \ k''[B]]$$

$$
\begin{array}{c}
(\nu w)(w[k[\underline{\mathsf{out}\ w}.\mathsf{in}\ k'.\mathsf{in}\ w.\mathbf{0}] | \mathsf{open}\ k'.\mathsf{open}\ k''.A]) \quad | \quad k'[\mathsf{open}\ k.k''[B]] \\
\downarrow \\
(\nu w)(w[\mathsf{open}\ k'.\mathsf{open}\ k''.A] \quad | \quad k[\underline{\mathsf{in}\ k'}.\mathsf{in}\ w.\mathbf{0}]) \quad | \quad k'[\mathsf{open}\ k.k''[B]] \\
\downarrow \\
(\nu w)(w[\mathsf{open}\ k'.\mathsf{open}\ k''.A] \quad | \quad k'[k[\mathsf{in}\ w.\mathbf{0}] | \underline{\mathsf{open}\ k}.k''[B]]) \\
\downarrow \\
(\nu w)(w[\mathsf{open}\ k'.\mathsf{open}\ k''.A] \quad | \quad k'[\underline{\mathsf{in}\ w}.\mathbf{0} | k''[B]]) \\
\downarrow \\
(\nu w)w[\underline{\mathsf{open}\ k'}.\mathsf{open}\ k''.A \ | \ k'[\mathbf{0} \ | \ k''[B]]] \\
\downarrow \\
(\nu w)w[\underline{\mathsf{open}\ k''}.A \ | \ k''[B]] \\
\downarrow \\
(\nu w)w[A \ | \ B]
\end{array}
$$

Locks and sums

$$acquire\ l.P \stackrel{\mathsf{def}}{=} \mathsf{open}\, l.P \qquad\qquad release\ l.P \stackrel{\mathsf{def}}{=} l[\,] \mid P$$

handshake: $acquire\ n.release\ m.P \mid release\ n.acquire\ m.Q$

Locks and sums

$$acquire\ l.P \stackrel{\mathsf{def}}{=} \mathsf{open}\, l.P \qquad\qquad release\ l.P \stackrel{\mathsf{def}}{=} l[\,] \mid P$$

handshake: $\quad acquire\ n.release\ m.P \mid release\ n.acquire\ m.Q$

$$\begin{aligned}[\![n \Rightarrow P \ + \ m \Rightarrow Q]\!] \stackrel{\mathsf{def}}{=} \ & (\nu p, q, r)\,(\\ & \quad p[\mathsf{in}\, n.\mathsf{out}\, n.q[\mathsf{out}\, p.\mathsf{open}\, r.P]] \mid \\ & \quad p[\mathsf{in}\, m.\mathsf{out}\, m.q[\mathsf{out}\, p.\mathsf{open}\, r.Q]] \mid \\ & \quad \mathsf{open}\, q \mid r[\,])\end{aligned}$$

$$[\![n \Rightarrow P \ + \ m \Rightarrow Q]\!] \mid n[R] \ \rightarrow^{*}\approx P \mid n[R]$$

Remarks

- mobility is the central mechanism

Remarks

- mobility is the central mechanism
- no substitution

Remarks

- mobility is the central mechanism
- no substitution
- ambients are trees that change structure over time

Remarks

- mobility is the central mechanism
- no substitution
- ambients are trees that change structure over time
- see an ambient as: $a[\ \underbrace{b_1[B_1]|\ldots|b_n[B_n]}_{\text{child ambients}}\ |\ \underbrace{P_1|\ldots|P_k}_{\text{active ``threads''}}\ ]$

Remarks

- mobility is the central mechanism
- no substitution
- ambients are trees that change structure over time
- see an ambient as: $a[\quad \underbrace{b_1[B_1]|\ldots|b_n[B_n]}_{\text{child ambients}} \quad | \quad \underbrace{P_1|\ldots|P_k}_{\text{active ``threads''}} \quad]$
- the model is "very distributed"
  - ▷ movement every where, at any level (except under capabilities)
  - ▷ autonomous movement: no synchronisation

Mobile Ambients are Turing complete

- ribbon cells: nested Ambients
  "leftmost and rightmost cells" : Ambients to extend the ribbon

- every cell contains an Ambient that represents its state

- head of the machine: **in** go right, **out** go left

Communications

what we have been discussing are *Pure Mobile Ambients*
let us now put some π-calculus into the language

Messages: $\langle M \rangle$ Abstractions: $(x).P$
Reduction: $\langle M \rangle \mid (x).P \;\to\; P_{\{x:=M\}}$

Communications

what we have been discussing are *Pure Mobile Ambients*
let us now put some π-calculus into the language

Messages: $\langle M \rangle$ Abstractions: $(x).P$
Reduction: $\langle M \rangle \mid (x).P \;\rightarrow\; P_{\{x:=M\}}$

- ▷ *local communications* (no medium)

Communications

what we have been discussing are *Pure Mobile Ambients*
let us now put some π-calculus into the language

Messages: $\langle M \rangle$ Abstractions: $(x).P$
Reduction: $\langle M \rangle \mid (x).P \;\rightarrow\; P_{\{x:=M\}}$

- *local communications* (no medium)
- communication of *capabilities* in a broad sense
  (name, capability, path)

Well-formed terms

$$P \stackrel{\text{def}}{=} (\nu n)\,P \mid \mathbf{0} \mid P_1|P_2 \mid\ !P \mid M[P] \mid M.P \mid (x).P \mid \langle M \rangle$$
$$M \stackrel{\text{def}}{=} x \mid n \mid \mathsf{in}\,M \mid \mathsf{out}\,M \mid \mathsf{open}\,M \mid \epsilon \mid M_1.M_2$$

- variables (x) stand for names and capabilities
- badly formed terms: $n.P$, $(\mathsf{in}\,n)[P]$
- a simple type system can be defined to rule out badly formed terms

π in MA

$$
\begin{aligned}
[\![(\nu n)\, P]\!] &\stackrel{\mathsf{def}}{=} (\nu n)\,(n[!\mathsf{open}\, io] \mid [\![P]\!]) \\
[\![n(x).P]\!] &\stackrel{\mathsf{def}}{=} (\nu p)\,(io[\mathsf{in}\, n.(x).p[\mathsf{out}\, n.[\![P]\!]]] \mid \mathsf{open}\, p) \\
[\![\overline{n}m]\!] &\stackrel{\mathsf{def}}{=} io[\mathsf{in}\, n.\langle m \rangle] \\
[\![P \mid Q]\!] &\stackrel{\mathsf{def}}{=} [\![P]\!] \mid [\![Q]\!] \qquad [\![!P]\!] \stackrel{\mathsf{def}}{=} ![\![P]\!]
\end{aligned}
$$

π in MA

$$
\begin{aligned}
[\![(\nu n)\, P]\!] &\stackrel{\mathsf{def}}{=} (\nu n)\,(n[!\mathsf{open}\, io] \mid [\![P]\!]) \\
[\![n(x).P]\!] &\stackrel{\mathsf{def}}{=} (\nu p)\,(io[\mathsf{in}\, n.(x).p[\mathsf{out}\, n.[\![P]\!]]] \mid \mathsf{open}\, p) \\
[\![\overline{n}m]\!] &\stackrel{\mathsf{def}}{=} io[\mathsf{in}\, n.\langle m\rangle] \\
[\![P \mid Q]\!] &\stackrel{\mathsf{def}}{=} [\![P]\!] \mid [\![Q]\!] \qquad [\![!P]\!] \stackrel{\mathsf{def}}{=} ![\![P]\!]
\end{aligned}
$$

$\longrightarrow$ *remote inputs and outputs*

Substitutions

- when adding communication, we brought back substitutions

Substitutions

- when adding communication, we brought back substitutions

- difference w.r.t. π: variable instanciation at any level
  or if you prefer: in π there is only one level

Substitutions

- when adding communication, we brought back substitutions
- difference w.r.t. π: variable instanciation at any level
  or if you prefer: in π there is only one level
- Zimmer 00 – π-calculus in Pure Mobile Ambients
  Mobile Ambients in Pure Mobile Ambients?

Substitutions

- when adding communication, we brought back substitutions
- difference w.r.t. π: variable instanciation at any level
  or if you prefer: in π there is only one level
- Zimmer 00 – π-calculus in Pure Mobile Ambients
  Mobile Ambients in Pure Mobile Ambients?
- similarly:
  are recursive definitions and replication equivalent in MA?

A young calculus

- since the 1998 paper *"Mobile Ambients"*, there have been numerous criticisms/counterproposals

the calculus, as it is, is:

- ▷ very difficult to implement

A young calculus

- since the 1998 paper *“Mobile Ambients”*, there have been numerous criticisms/counterproposals

the calculus, as it is, is:

- ▷ very difficult to implement
- ▷ very difficult to reason about

A young calculus

- since the 1998 paper *"Mobile Ambients"*, there have been numerous criticisms/counterproposals

the calculus, as it is, is:

- ▷ very difficult to implement
- ▷ very difficult to reason about

barbed equivalence: barbs are *ambients*

A young calculus

- since the 1998 paper *"Mobile Ambients"*, there have been numerous criticisms/counterproposals

the calculus, as it is, is:

- ▷ very difficult to implement
- ▷ very difficult to reason about

barbed equivalence: barbs are *ambients*
but then, what about a labelled transition system?