Mobile processes & types, part II

Davide Sangiorgi

University of Bologna

Email: davide.sangiorgi@cs.unibo.it
http://www.cs.unibo.it/~sangio/

June 25, 2003

Part II: Distributed mobile processes

CONTENTS

- **Name mobility**: Processes communicate via names (channels) and names may move (π-calculus)

- **Process mobility**: Processes communicate via names and processes may move (HOπ, Pict, ...)

- **Mobility on distributed systems**: Computation is local to sites (also called locations) and processes may move between sites

Process mobility may be reduced to name mobility

For distributed systems, the reduction to name mobility is less clear, and less useful

Distributed mobile computing

A new paradigm for structuring distributed systems.

Software ‘pieces’ migrate from remote sites to a host, on a Wide Area Network such as Internet, and interact with the resources and facilities local to that host.

Not a new idea. Examples: Postscript, Database SQL queries.

Important issues:

- heterogenenous networks
- protection domains and firewalls
- bandwidth fluctuation
- failures

Motivations

- Real-time interaction with remote resources (example: allowing embedded executable contents to remove the constraints of HTML and network latency)

- Reduction of Communication Traffic (efficiency)

- Customization and extension of server capabilities (flexibility)

- Avoiding distribution of state (storage)

Some languages

Java (Sun)

Telescript and Odyssey (General Magic)

Obliq (Cardelli)

Voyager (Object Space)

Aglets (IBM)

...

Distributed Join (INRIA Rocq.)

Nomadic Pict (Cambridge)

Ambit (Cardelli)

Klava (Florence)

...

Some calculi

π-calculus

Dπ (Hennessy and Riely)

Ambients (Cardelli and Gordon)

Seal (Castagna and Vitek)

Klaim (Florence group)

...

A huge number, mostly extensions of π-calculus or related calculi

Both the languages and the models for distributed mobility are pretty new. It is therefore difficult to make assessments.
We can however try to isolate the important dimensions and examine their implications.

Fundamental design decisions

- **What is movement?**

  code mobility (only the code of a process can move)

  agent mobility (both code and state of a process can move)

- **What is the structure of locations?**

  flat domain

  hierarchical domain

- **What is communication?**

  global (the links, references, etc. that a process has remain intact when the process moves)

  local (a process may only use links, references etc. to communicate with processes in surrounding locations)

Other issues

- Communication: several variations possible
  - For local communication, should commmunications among processes in parent and child locations be allowed? and in siblings locations?
  - what are transmittable values? names, processes, ...
  - how is communication achieved? asynchronous, synchronous, broadcast, ...
- security policy offered
- type system
  - as code may be downloaded, it may be impossible to guarantee the absence of type-errors statically
  - proof carryng code
- scoping
- (for languages) translation strategy
- (for languages) dynamic code linking (the “code on demand” approach)

Ambients

Extends the π-calculus with

- agent mobility
- hierarchical locations
- local communications

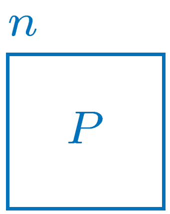
CONTENTS

Ambients

[Cardelli,Gordon'98]

An ambient is a named location:

Graphical representation:



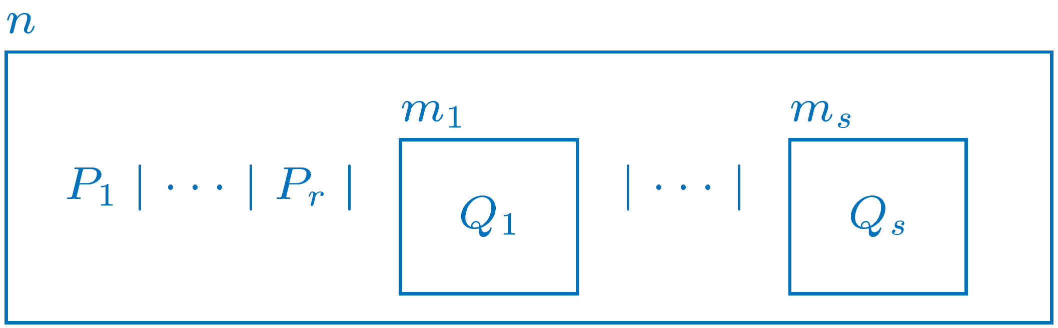
Syntax: $n[\, P \,]$

- n is the name of the ambient
- P is the process running inside it

P is active in $n[\, P \,]$:

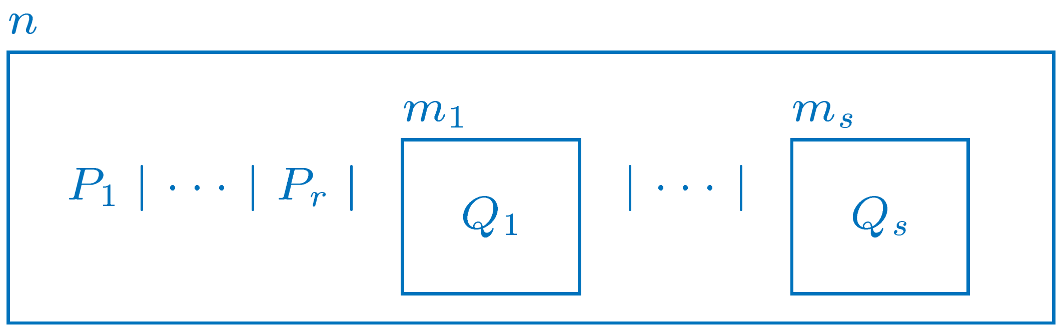
$$P \longrightarrow P' \Rightarrow n[\, P \,] \longrightarrow n[\, P' \,]$$

An ambient can contain other ambients and local processes:



– $m_i[\, Q_i \,]$: a *child ambient*

Ambients can move in and out of other ambients, and can be opened.



where each P_i is not an ambient or a parallel composition

- P_i: a *local process*

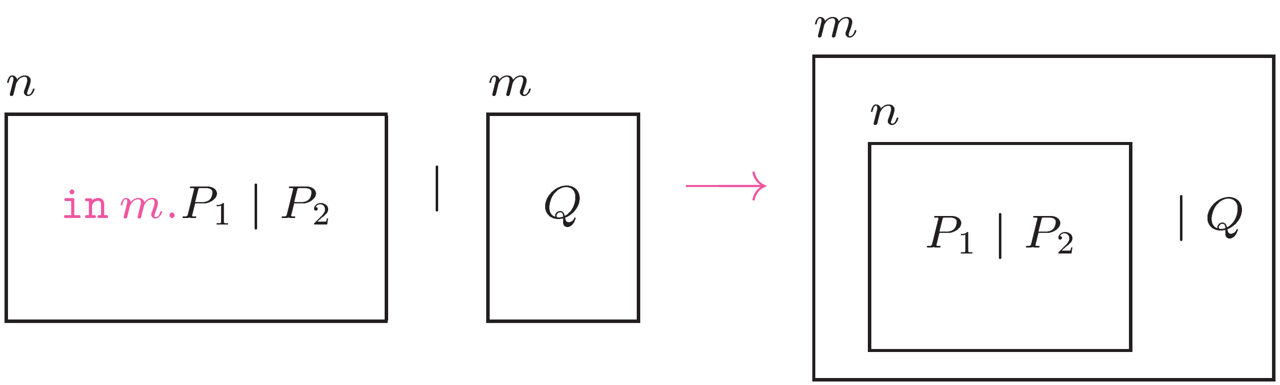
The ambients movements are determined by the capabilities exercised by the local processes

$M \,.\, P$ can execute capability M and then continues as P

(as in the π-calculus, '.' represents sequentialisation)

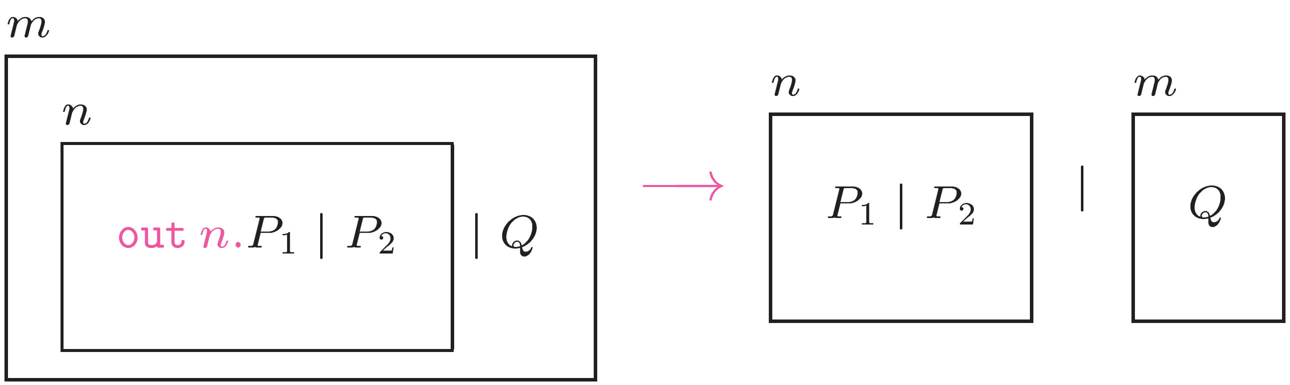
3 forms of capabilities

The $\mathtt{in}$ movement



$$n[\,\mathtt{in}\ m.P_1 \mid P_2\,] \mid m[\,Q\,] \longrightarrow m[\,n[\,P_1 \mid P_2\,] \mid Q\,]$$

The `out` movement



$$m[\, n[\, \mathtt{out}\ m.P_1 \mid P_2 \,] \mid Q \,] \longrightarrow n[\, P_1 \mid P_2 \,] \mid m[\, Q \,]$$

The `open` movement

$$\texttt{open}\, n.P \mid \overset{n}{\boxed{Q}} \longrightarrow P \mid Q$$

$$\texttt{open}\, n.P \mid n[\, Q \,] \longrightarrow P \mid Q$$

... potentially very upsetting!

Communications

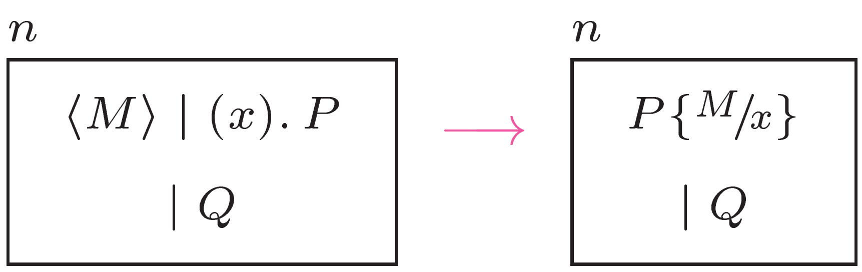
Local processes of an ambient can be messages and abstractions (= output, input)

$$\langle M \rangle \mid (x).\, P \longrightarrow P\{^{M}\!/\!_{x}\}$$

Cmmunication is *local* to ambients.

Example:

$$n[\, Q \mid \langle M \rangle \mid (x).\, P \,] \longrightarrow n[\, Q \mid P\{^{M}\!/\!_{x}\} \,]$$



Ambients: syntax

$n, m..$ *Names*

$x, y..$ *Variables*

			Processes				*Capabilities*
P	$::=$	$M[\,P\,]$	ambient	M	$::=$	n	name
	$\mid$	$M.\,P$	action		$\mid$	x	variable
	$\mid$	$\mathbf{0}$	nil process		$\mid$	$\mathtt{in}\ M$	can enter M
	$\mid$	$P \mid P$	parallel		$\mid$	$\mathtt{out}\ M$	can exit M
	$\mid$	$\boldsymbol{\nu} n\ P$	restriction		$\mid$	$\mathtt{open}\ M$	can open M
	$\mid$	$!P$	replication (or recursion)		$\mid$	$M_1.\,M_2$	path
	$\mid$	$\langle M \rangle$	message (= output)		$\mid$	ϵ	empty path
	$\mid$	$(x).\,P$	abstraction (= input)				

Reduction semantics

Structural congruence: the expected rules, plus

$$
\begin{array}{rcl}
\boldsymbol{\nu} n\; m[\, P \,] & \equiv & m[\, \boldsymbol{\nu} n\; P \,] \\
\epsilon.\, P & \equiv & P \\
(M.\, M').\, P & \equiv & M.\, (M'.\, P)
\end{array}
$$

Inference rules for reduction:

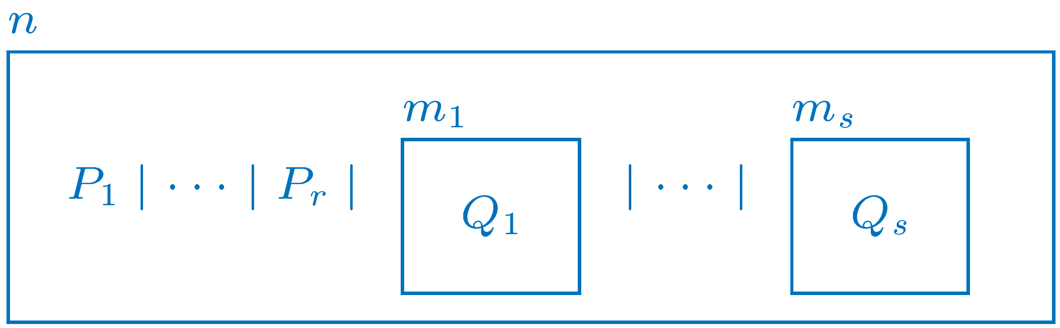
$$
\frac{P_1 \longrightarrow P_1'}{P_1 \mid P_2 \longrightarrow P_1' \mid P_2}
\qquad
\frac{P \longrightarrow P'}{\boldsymbol{\nu} z\; P \longrightarrow \boldsymbol{\nu} z\; P'}
$$

$$
\frac{P \longrightarrow P'}{n[\, P \,] \longrightarrow n[\, P' \,]}
\qquad
\frac{P_1 \equiv P_2 \longrightarrow P_2' \equiv P_1'}{P_1 \longrightarrow P_1'}
$$

$\Longrightarrow \;\triangleq\;$ the reflexive and transitive closure of $\longrightarrow$

Labelled transtion system semantics: much harder [Merro and Hennessy 2002]

Normal form of an ambient:



where each P_i is not an ambient, a parallel composition, or a restriction, and can therefore only be:

- a capability-prefixed process, $M.\,P$
- an abstraction $(x).\,P$
- a message, $\langle M \rangle$
- a replication

Alternative primitives, I: Subjective vs objective moves

Subjective: I move (from the inside)

$$n[\, \mathtt{in}\ m.P_1 \mid P_2\,] \mid m[\, Q\,] \quad\longrightarrow\quad m[\, n[\, P_1 \mid P_2\,] \mid Q\,]$$

$$m[\, n[\, \mathtt{out}\ m.P_1 \mid P_2\,] \mid Q\,] \quad\longrightarrow\quad n[\, P_1 \mid P_2\,] \mid m[\, Q\,]$$

Objective: I make you move (from the outside)

$$\mathtt{mv}\ \mathtt{in}\ m.\ P \mid m[\, Q\,] \quad\longrightarrow\quad m[\, P \mid Q\,]$$

$$m[\, \mathtt{mv}\ \mathtt{out}\ m.\ P \mid R\,] \quad\longrightarrow\quad R \mid m[\, R\,]$$

(less powerful primitives)

$$\mathtt{mv}\ n\ \mathtt{in}\ m.\ P \mid n[\, R\,] \mid m[\, Q\,] \quad\longrightarrow\quad P \mid m[\, n[\, R\,] \mid Q\,]$$

$$m[\, \mathtt{mv}\ n\ \mathtt{out}\ m.\ P \mid n[\, R\,] \mid Q\,] \quad\longrightarrow\quad n[\, R\,] \mid m[\, P \mid Q\,]$$

(more complicated primitives)

Alternative primitives, II: Open vs Acid moves

Acid move: "I dissolve my own boundary"

$$n[\,\texttt{acid}\,.\,P \mid Q\,] \longrightarrow P \mid Q$$

Thus objective moves are derivable (example; the others are an exercise):

$$\texttt{mv in}\ n.\,P \ \triangleq\ \boldsymbol{\nu} q\ q[\,\texttt{in}\,n.\,\texttt{acid}\,.\,P\,]$$

This gives a kidnapping ability:

$$\begin{array}{rcl} \texttt{entrap}\,m & \triangleq & \boldsymbol{\nu} k\ (k[\ ] \mid \texttt{mv in}\ m.\,\texttt{in}\,k) \\ \texttt{entrap}\,m \mid m[\,P\,] & \Longrightarrow & \boldsymbol{\nu} k\ k[\,m[\,P\,]\,] \end{array}$$

(which also shows that it is good that objective moves cannot always be simulated)

$\boxed{\text{the implications of (combinations of) the primitives has to be carefully considered, notably in terms of security}}$

CONTENTS

Locks as Ambients

A lock called n becomes an ambient $n[\ ]$

Lock operations:

$$\begin{aligned} \mathtt{release}\, n.\, P &\triangleq n[\ ] \mid P \\ \mathtt{getlock}\, n.\, P &\triangleq \mathtt{open}\, n.\, P \end{aligned}$$

Message delivery

Deliver message M from location n to process P at location m:

$$n : M \Longrightarrow n : P$$

$$
\begin{aligned}
& n[\, Q \mid p[\, \mathtt{out}\; n.\mathtt{in}\; m \mid \langle M \rangle \,]\,] \mid m[\, \mathtt{open}\; p.\,(x).\, P \,] \\
\longrightarrow\quad & n[\, Q \,] \mid p[\, \mathtt{in}\; m \mid \langle M \rangle \,] \mid m[\, \mathtt{open}\; p.\,(x).\, P \,] \\
\longrightarrow\quad & n[\, Q \,] \mid m[\, p[\, \langle M \rangle \,] \mid \mathtt{open}\; p.(x).\, P \,] \\
\longrightarrow\quad & n[\, Q \,] \mid m[\, \langle M \rangle \mid (x).\, P \,] \\
\longrightarrow\quad & n[\, Q \,] \mid m[\, P\{^{M}\!/\!_{x}\} \,]
\end{aligned}
$$

Renaming

$$n \,\mathsf{be}\, m.\, P \;\triangleq\; m[\, \mathtt{out}\; n.\, \mathtt{open}\; n.\, P \,] \mid \mathtt{in}\; m$$

$$\begin{aligned}
& n[\, n \,\mathsf{be}\, m.\, P \mid Q \,] = n[\, m[\, \mathtt{out}\; n.\, \mathtt{open}\; n.\, P \,] \mid \mathtt{in}\; m \mid Q \,] \\
\longrightarrow\quad & m[\, \mathtt{open}\; n.\, P \,] \mid n[\, \mathtt{in}\; m \mid Q \,] \\
\longrightarrow\quad & m[\, \mathtt{open}\; n.\, P \mid n[\, Q \,]\,] \\
\longrightarrow\quad & m[\, P \mid Q \,]
\end{aligned}$$

Mimicking objective moves

Objective enter:

$$\begin{array}{rcl}
n^{\downarrow}[\, P \,] & \triangleq & n[\, P \mid \texttt{allow}\ in \,] \\
\texttt{allow}\ n & \triangleq & !\texttt{open}\ n \\
\texttt{mv in}\ n.\, P & \triangleq & \boldsymbol{\nu} k\ k[\, \texttt{in}\ n.\, in[\, \texttt{out}\ k.\, P \,]\,]
\end{array}$$

We have:

$$\begin{array}{rcl}
\texttt{mv in}\ n.\, P \mid n^{\downarrow}[\, Q \,] & \longrightarrow & n^{\downarrow}[\, \boldsymbol{\nu} k\ k[\, in[\, \texttt{out}\ k.\, P \,]\,] \mid Q \,] \\
& \longrightarrow & n^{\downarrow}[\, \boldsymbol{\nu} k\ k[\ ] \mid in[\, P \,] \mid Q \,] \\
& \longrightarrow & n^{\downarrow}[\, \boldsymbol{\nu} k\ k[\ ] \mid P \mid Q \,] \\
& \approx & n^{\downarrow}[\, P \mid Q \,]
\end{array}$$

Objecive exit:

$$\begin{array}{rcl} n^{\uparrow}[\ ] & \triangleq & n[\,P\,] \mid \texttt{allow}\ out \\ \texttt{mv out}\ n.\,P & \triangleq & \boldsymbol{\nu} k\ k[\,\texttt{out}\ n.\,out[\,\texttt{out}\ k.\,P\,]\,] \end{array}$$

We have:

$$n^{\uparrow}[\,\texttt{mv out}\ n.\,P \mid Q\,] \implies\approx \quad P \mid n^{\uparrow}[\,Q\,]$$

Routable Packets

A packet is repeatedly routed to various destinations:

$$PKT \triangleq pkt[\,!(x).\,x \mid !\mathtt{open}\,route\,] \qquad \text{(the packet)}$$

$$\langle P, M\rangle \triangleq route[\,\mathtt{in}\,pkt.\,\langle M\rangle \mid P\,] \qquad (P \text{ is routed to destination } M\,)$$

We have:

$$PKT \mid \langle P_1, \mathtt{in}\,n.\,\mathtt{in}\,k\rangle \mid n[\,Q_1 \mid m[\,Q_2\,] \mid k[\,Q_3\,]\,] \mid q[\,Q_2\,].$$

$$\Longrightarrow \quad n[\,Q_1 \mid m[\,Q_2\,] \mid k[\,PKT \mid P_1\,]\,]$$

Crossing a firewall, I

Assume the shared key k is known to the firewall and the client:

$$
\begin{array}{lcl}
FW & \triangleq & (\boldsymbol{\nu} w, r)\ (\langle \mathtt{in}\, r \rangle \mid r[\ \mathtt{open}\, k.\ \mathtt{in}\, w\ ] \mid w[\ \mathtt{open}\, r.\ P\ ]) \\
AG & \triangleq & (x).\ k[\ x.\ C\ ]
\end{array}
$$

$$
\begin{array}{ll}
 & AG \mid FW \\
\equiv & (\boldsymbol{\nu} w, r)\ ((x).\ k[\ x.\ C\ ] \mid \langle \mathtt{in}\, r \rangle \mid r[\ \mathtt{open}\, k.\ \mathtt{in}\, w\ ] \mid w[\ \mathtt{open}\, r.\ P\ ]) \\
\Longrightarrow & (\boldsymbol{\nu} w, r)\ (k[\ \mathtt{in}\, r.\ C\ ] \mid r[\ \mathtt{open}\, k.\ \mathtt{in}\, w\ ] \mid w[\ \mathtt{open}\, r.\ P\ ]) \\
\Longrightarrow & (\boldsymbol{\nu} w, r)\ (r[\ k[\ C\ ] \mid \mathtt{open}\, k.\ \mathtt{in}\, w\ ] \mid w[\ \mathtt{open}\, r.\ P\ ]) \\
\Longrightarrow & (\boldsymbol{\nu} w, r)\ (r[\ C \mid \mathtt{in}\, w\ ] \mid w[\ \mathtt{open}\, r.\ P\ ]) \\
\Longrightarrow & (\boldsymbol{\nu} w, r)\ (w[\ r[\ C\ ] \mid \mathtt{open}\, r.\ P\ ]) \\
\Longrightarrow & (\boldsymbol{\nu} w)\ w[\ C \mid P\ ]
\end{array}
$$

A problem: the message $\mathtt{in}\, r$ could be stolen by a malicious ambient and duplicated. This ambient could then get inside the firewall

Crossing a firewall, II

The firewall ambient sends into the agent a pilot ambient k with the capability $\mathtt{in}\ w$ for entering the firewall.

$$
\begin{array}{lcl}
FW & \triangleq & \boldsymbol{\nu} w\ (w[\ k[\ \mathtt{out}\ w.\ \mathtt{in}\ k.\ \mathtt{in}\ w\ ] \mid \mathtt{open}\ k.\ P\ ]) \\
AG & \triangleq & k[\ \mathtt{open}\ k.\ C\ ]
\end{array}
$$

$$
\begin{array}{ll}
& AG \mid FW \\
\Longrightarrow & \boldsymbol{\nu} w\ (k[\ \mathtt{open}\ k.\ C\ ] \mid k[\ \mathtt{in}\ k.\ \mathtt{in}\ w\ ] \mid w[\ \mathtt{open}\ k.\ P\ ] \\
\Longrightarrow & \boldsymbol{\nu} w\ (k[\ k[\ \mathtt{in}\ w\ ] \mid \mathtt{open}\ k.\ C\ ] \mid w[\ \mathtt{open}\ k.\ P\ ] \\
\Longrightarrow & \boldsymbol{\nu} w\ (k[\ \mathtt{in}\ w \mid \mathtt{open}\ k.\ C\ ] \mid w[\ \mathtt{open}\ k.\ P\ ] \\
\Longrightarrow & \boldsymbol{\nu} w\ w[\ k[\ C\ ] \mid \mathtt{open}\ k.\ P\ ] \\
\Longrightarrow & \boldsymbol{\nu} w\ w[\ C \mid P\ ]
\end{array}
$$

A problem: movements initiated by C could disrupt the protocol

Crossing a firewall, III

Names m, k, k' act like passwords which guarantee the access only to authorized agents.

$$\begin{array}{lcl} FW & \triangleq & \boldsymbol{\nu} w\,(w[\,\mathtt{open}\,m.\,\mathtt{open}\,k'.\,P \mid k[\,\mathtt{out}\,w.\,\mathtt{in}\,m.\,\mathtt{in}\,w\,]\,]) \\ AG & \triangleq & m[\,\mathtt{open}\,k.\,k'[\,Q\,]\,] \end{array}$$

Correctness of the protocol:

$$(\boldsymbol{\nu} m, k, k'\,)(FW \mid AG) \approx \boldsymbol{\nu} w\,(w[\,Q \mid P\,])$$

under the conditions that $w \notin \mathrm{fn}(Q)$ and $(\mathrm{fn}(Q) \cup \mathrm{fn}(P)) \cap \{m, k, k'\} = \emptyset$.

The “perfect firewall equation”

$$\boldsymbol{\nu} n \; n[\, P \,] \approx \mathbf{0} \qquad n \text{ not free in } P$$

- $n[\, P \,]$ is a firewall named n surrounding P
- $\mathtt{in}\; n$ is the capability of entering the firewall
- $\mathtt{out}\; n$ is the capability needed to exit the firewall

Encoding π-calculus (ie, channels) into Ambients

For the translation of a name p, we use two names: $p\mathtt{CH}$, $p\mathtt{IO}$

$$
\begin{array}{rcl}
\{\![\, p(q_1 .\, .\, q_n).\, P \,]\!\} & \triangleq & \boldsymbol{\nu} k \;(\;\; p\mathtt{IO}[\;\; \mathtt{in}\, p\mathtt{CH}.\, (q_1\mathtt{CH}, q_1\mathtt{IO} .\, .\, q_n\mathtt{CH}, q_n\mathtt{IO}).\;\; k[\, \mathtt{out}\, p\mathtt{CH}.\, \{\![\, P \,]\!\} \,]\;\;] \\
& & \qquad \mid \mathtt{open}\, k\,)
\end{array}
$$

$$
\begin{array}{rcl}
\{\![\, \overline{p}\langle q_1 .\, .\, q_n \rangle \,]\!\} & \triangleq & p\mathtt{IO}[\, \mathtt{in}\, p\mathtt{CH}.\, \langle q_1\mathtt{CH}, q_1\mathtt{IO} .\, .\, q_n\mathtt{CH}, q_n\mathtt{IO} \rangle \,] \\
\{\![\, \boldsymbol{\nu} p \; P \,]\!\} & \triangleq & \boldsymbol{\nu} p\mathtt{CH}\, \boldsymbol{\nu} p\mathtt{IO}\, (p\mathtt{CH}[\, !\mathtt{open}\, p\mathtt{IO} \,] \mid \{\![\, P \,]\!\}) \\
\{\![\, P \mid Q \,]\!\} & \triangleq & \{\![\, P \,]\!\} \mid \{\![\, Q \,]\!\} \\
\{\![\, !P \,]\!\} & \triangleq & !\{\![\, P \,]\!\} \\
\{\![\, \mathbf{0} \,]\!\} & \triangleq & \mathbf{0}
\end{array}
$$

The complete encoding needs further encoding step, to add channel ambients for all the free names of a process:

$$\{[P]\} \quad \triangleq \quad \{[\ P\ ]\} \mid p_1\mathtt{CH}[\ !\mathtt{open}\, p_1\mathtt{IO}\ ] \mid \ldots \mid p_n\mathtt{CH}[\ !\mathtt{open}\, p_n\mathtt{IO}\ ]$$

where $p_1 \ldots p_n$ are free names of P

CONTENTS

A simpler calculus: MA$^-$

No communication of capabilities

P	$::=$	$p[\,P\,]$	ambient
	$\mid$	$M\,.\,P$	action
	$\mid$	$\mathbf{0}$	nil process
	$\mid$	$P \mid P$	parallel
	$\mid$	$\boldsymbol{\nu} n\; P$	restriction
	$\mid$	$!P$	replication (or recursion)
	$\mid$	$\langle p \rangle$	message (= output)
	$\mid$	$(x)\,.\,P$	abstraction (= input)

p	$::=$	n	name
	$\mid$	x	variable
M	$::=$	$\mathtt{in}\,p$	can enter n
	$\mid$	$\mathtt{out}\,p$	can exit n
	$\mid$	$\mathtt{open}\,p$	can open n
	$\mid$	$M_1\,.\,M_2$	path
	$\mid$	ϵ	empty path

Communication of capabilities can sometimes be simulated:

$$n[\, Q \mid \langle \mathtt{in}\, m \rangle \mid (x).\, x.\, R \,] \longrightarrow n[\, Q \mid \mathtt{in}\, m.\, R \,]$$

becomes

$$\begin{array}{rcl} n[\, Q \mid (\boldsymbol{\nu} k, h)\, (k[\, \mathtt{in}\, m.\, h[\ ]\,] \mid \mathtt{open}\, k.\, \mathtt{open}\, h.\, R)\,] & \longrightarrow\approx & n[\, Q \mid \mathtt{in}\, m.\, \boldsymbol{\nu} h\, (h[\ ] \mid \mathtt{open}\, h.\, R)\,] \\ & \approx & n[\, Q \mid \mathtt{in}\, m.\, R \,] \end{array}$$

Run-time errors in MA$^-$

Disagreements on the kind of values exchanged within an ambient:

$$n[\ \langle \mathtt{true} \rangle \mid (x).\ \ \mathtt{if}\ \ x = 0\ \ \mathtt{then}\ \ P\ \ \mathtt{else}\ \ Q\ ]$$

$$n[\ m[\ \langle \mathtt{true} \rangle\ ] \mid \mathtt{open}\, m \mid \langle 5 \rangle \mid (x).\ \ \mathtt{if}\ \ x = 0\ \ \mathtt{then}\ \ P\ \ \mathtt{else}\ \ Q\ ]$$

We assign a type to each ambient, to specify the type of the values that can be exchanged within that ambient

Type grammar

T	$:=$	$\mathtt{Int} \;\big|\; \mathtt{Bool} \;\big|\; \ldots$	integers, booleans ...
	$\big|$	$\mathtt{Amb}\,[T]$	an ambient inside which messages have type T
	$\big|$	$\mathtt{Shh}$	no exchange

– An ambient with no messages: $\mathtt{Amb}\,[\mathtt{Shh}]$

– An ambient with integer messages: $\mathtt{Amb}\,[\mathtt{Int}]$

– An ambient where messages are names of quite ambients: $\mathtt{Amb}\,[\mathtt{Amb}\,[\mathtt{Shh}]]$

– $\Gamma \vdash P : T$ reads "under the type assumptions Γ for the names of ambients, P may unleash messages of type T"

Examples

$$
\begin{array}{rcl}
n : \mathtt{Amb}\,[\mathtt{Shh}], m : \mathtt{Amb}\,[T] & \vdash & n[\,\mathtt{in}\,m\,] \\
 & \nvdash & n[\,\langle M\rangle\,]
\end{array}
$$

$$
m : \mathtt{Amb}\,[\mathtt{Int}], n : \mathtt{Amb}\,[\mathtt{Int}] \quad \vdash \quad n[\,\langle 5\rangle \mid m[\,\langle 3\rangle\,] \mid \mathtt{open}\,m\,]
$$

Analogy with types for π

An ambient $\Rightarrow$ a channel ether

$$\texttt{Amb}\,[T] \quad \Rightarrow \quad \sharp\, T$$

Following this intuition, the typing rules are as expected

Example rules

$$\frac{\Gamma \vdash P_1 \;:\; T \qquad \Gamma \vdash P_2 \;:\; T}{\Gamma \vdash P_1 \mid P_2 \;:\; T} \qquad \frac{\Gamma \vdash M \;:\; T \qquad \Gamma \vdash P \;:\; T}{\Gamma \vdash M.\,P \;:\; T}$$

$$\frac{\Gamma \vdash M \;:\; T}{\Gamma \vdash \langle M \rangle \;:\; T} \qquad \frac{\Gamma, x : T \vdash P \;:\; T}{\Gamma \vdash (x : T)P \;:\; T}\; T \neq \mathtt{Shh}$$

$$\frac{\Gamma \vdash p \;:\; \mathtt{Amb}\,[T]}{\Gamma \vdash \mathsf{open}\, p \;:\; T}$$

Since $\mathbf{0}$ and $n[\, P\,]$ unleash nothing at the current level, they can have any type:

$$\frac{\Gamma \vdash m \;:\; \mathtt{Amb}\,[S] \qquad \Gamma \vdash P \;:\; S}{\Gamma \vdash p[\, P\,] \;:\; T} \qquad \frac{}{\Gamma \vdash \mathbf{0} \;:\; T}$$

Typing MA

[Cardelli, Gordon, Ghelli, ... 98-...]

An additional problem: unlike MA$^-$ (or the π-calculus) untyped MA contains garbage terms

$$\texttt{out}\; m[\, P \,] \qquad\qquad\qquad n.\, P$$

They can appear after some reductions:

$$\langle \texttt{out}\; m \rangle \mid (x).\, x[\, P \,] \longrightarrow \texttt{out}\; m[\, P \,]$$

$$\langle n \rangle \mid (x).\, x[\, P \,] \longrightarrow n.\, P$$

We have to add another form of types, to distinguish capabilities.

A type `Cap` is not enough: using a capability may unleash messages:

$$\langle \mathtt{open}\, n \rangle \mid (x).\, x \mid n[\, \langle 5 \rangle \,]$$

$\mathtt{Cap}\,[T]$: the type of a capability that, when used, may unleash messages of type T

Example:

$$m : \mathtt{Amb}\,[\mathtt{Cap}\,[\mathtt{Int}]], n : \mathtt{Amb}\,[\mathtt{Int}] \vdash m[\, \langle \mathtt{open}\, m \rangle \mid (x).\, x \mid n[\, \langle 5 \rangle \,]\,]$$

- New typing rules for capabilities

$$\frac{\Gamma \vdash M \ : \ \texttt{Cap}\,[T] \qquad \Gamma \vdash P \ : \ T}{\Gamma \vdash M.P \ : \ T}$$

$$\frac{\Gamma \vdash M \ : \ \texttt{Amb}\,[T]}{\Gamma \vdash \texttt{open}\, M \ : \ \texttt{Cap}\,[T]} \qquad \frac{\Gamma \vdash M \ : \ \texttt{Amb}\,[T]}{\Gamma \vdash \texttt{in}\, M \ : \ \texttt{Cap}\,[S]}$$

- Usual subject reduction theorems

Types for controlling mobility

- An effect is something important a process can do

- Interpretations:

$$
\begin{array}{rcl}
\Gamma \vdash P : E & \triangleq & P \text{ has at most effect } E \\
n : \mathtt{Amb}[E] & \triangleq & \text{the effect of running } n \text{ is at most } E \\
M : \mathtt{Cap}[E] & \triangleq & \text{the effect of using } M \text{ is at most } E
\end{array}
$$

- Some examples:

– previous slides: effects are input/output operations

– following slides: effects are also in/out/open movements

Tracking in/out movements

Separate:

- *mobile* ambients (can move, exercising in/out capabilities)
- *immobile* ambients

$$\begin{array}{lcl} E & ::= & Z, T \\ Z & ::= & \mathtt{Mob} \;\big|\; \mathtt{Imm} \end{array}$$

$\mathtt{Amb}\,[\mathtt{Mob}, \mathtt{Shh}] \triangleq$ a quite ambient that can move

$\mathtt{Imm}, \mathtt{Int} \triangleq$ a process that can unleash integers and cannot exercise in/out capabilities

Example rules

$$\frac{\Gamma \vdash M \;:\; \mathtt{Cap}\,[Z,T] \qquad \Gamma \vdash P \;:\; Z,T}{\Gamma \vdash M\,.\,P \;:\; Z,T} \qquad \frac{\Gamma \vdash M \;:\; T}{\Gamma \vdash \langle M \rangle \;:\; Z,T}$$

$$\frac{\Gamma \vdash M \;:\; \mathtt{Amb}\,[E]}{\Gamma \vdash \mathtt{in}\,M \;:\; \mathtt{Cap}\,[\mathtt{Mob}, S]}$$

Groups

Adapting π-calculus sorting system **[Milner 91]**

We partition ambients into groups

We can thus say that certain ambients can only cross ambients of certain groups

$$\begin{array}{rcll} H, G & & & \text{groups} \\ Hs & ::= & \{H_1 \ldots H_n\} & \text{set of groups} \\ E & ::= & Hs, T & \text{effect} \end{array}$$

$n : \mathtt{G}\,[H, \mathtt{Int}] \quad \triangleq \quad$ n belongs to group G,
inside n integers can be exchanged,
n can only enter ambients of group H

(H could be a group of trusted ambients)

$\Gamma \vdash P \; : \; \{H_1, H_2\}, \mathtt{Bool} \quad \triangleq \quad$ P can unleash boolean messages,
and enter ambients in the groups H_1, H_2

NB: To be more precise indicating names of ambients crossed (such as: ambient n only enter ambient m): we would need dependent types (much more complex)

Further extensions

- Control of openings

$$E ::= {}^{o}H, T \quad \text{unleash messages of type } T\text{, can open ambients of group } H$$

The transitive closure of openings must be present in the type

Example: Given

$$\mathtt{G}_1 \, [{}^{o}Hs_1, T_1] \; , \qquad \mathtt{G}_2 \, [{}^{o}Hs_2, T_2]$$

with $G_2 \in Hs_1$, then $Hs_2 \subseteq Hs_1$

- Dynamic creation of groups

CONTENTS

Rational behind Ambients

- Everything is a barrier

- phisical locations
- firewalls
- security issues
  - reduced to the capability of entering, exiting, destroying, creating barriers

- Mobility is movement of barriers

- Failures are not observable

- site failure is not the only reason why a communication may not be immediately possible

Rational behind Ambients (cntd)

- Communication is local within a barrier (in practice, an ambient may contain familiar programming abstractions such as objects and threads)

- Reliable remote communications do not exist

- simulated by sending mobile packets
- separate the *capability* to communicate from the *ability* to communicate.
  - capability does not imply ability. Example: congestion, security, ...

Some debatable aspects

- interferences
- `open` primitive
- implementations
- operational and algebraic theory

The `open` primitive

Opening an ambient from the outside is potentially very dangerous

Opening from the inside is more dangerous (cf: acid move)

In favour: $\mathtt{open}\ n$ is a capability that is given out by n; so $n[\,P\,]$ is not dissolved if it does not wish to be

Against: sometimes ambient should indeed be dissolved, but only after some task has been completed. This is hard to control with the `open` primitive.

Distributed implementations of ambient-like languages

- each movement operation involves ambients on different hierarchical levels, and may involve severla ambients at of same level:

$$\begin{array}{rl} & m[\, n[\, \mathtt{out}\, m.\, Q \mid \mathtt{in}\, p.\, R\,] \mid p[\, \mathtt{in}\, n.\, S\,]\,] \\ \mathtt{open}\, m \mid & m[\, n[\, \mathtt{out}\, m.\, Q \mid \mathtt{in}\, p.\, R\,] \mid p[\, \mathtt{in}\, n.\, S\,]\,] \end{array}$$

This requires a **synchronisation** among all ambients affected by a movement.

(cf: locks **[Cardelli]**,
protocols of asynchronous messages **[Fournet, Schmitt, Lévy]**)

CONTENTS

Interference

- Interference:

  The activity of a process is damaged or corrupted because of the activities of other processes.

- Interference is one of the most tricky and subtle phenomena in concurrency.

  It can make reasoning on programs very hard.

- The most important form of interference in process calculi is determined by **redex overlapping**.

Plain interferences

A process may engage **the "same" interaction** with two different partners.

Examples (in Ambients):

$$\mathtt{open}\, n.\, P \mid \mathtt{open}\, n.\, Q \mid n[\, R\,]$$

$$m[\, \mathtt{in}\, n.\, P\,] \mid n[\, Q\,] \mid n[\, R\,]$$

Similar to the interferences of the π-calculus:

$$x(y).\, P \mid \overline{x}\, v.\, Q \mid \overline{x}\, w.\, R$$

Plain interferences sometimes allow us to program useful non-deterministic behaviours.

Grave interferences

In the π-calculus all interferences are plain interferences

In Ambients, other forms of interference exist: the **grave interferences**. Examples:

$$\texttt{open}\, n \mid n[\, \texttt{in}\, m.\, P \,] \mid m[\, Q \,]$$

$$h[\, n[\, \texttt{in}\, m.\, P \,] \mid \texttt{out}\, h.\, R \,] \mid m[\, Q \,]$$

$$h[\, P \,] \mid n[\, \texttt{in}\, h \mid m[\, \texttt{out}\, n.\, Q \,]\,]$$

These systems are, from n's point of view, inconsistent.

- A grave interference evidences a **logical conflict** on how a given ambient is supposed to evolve.
- Grave interferences represent **programming errors**.

The coexistence of plain and grave interferences has consequences on:

- **algebraic theory**
- **programming**
- **proofs of behavioural properties**
- **implementation**
- **types and static analysis techniques**

Finding conditions on Ambients that guarantee absence of grave interferences appears hard.

Example: interferences for the `in` capability

$$m[\, \mathtt{in}\, n.\, P_1 \mid P_2 \,] \mid n[\, Q \,] \longrightarrow n[\, Q \mid m[\, P_1 \mid P_2 \,]\,]$$

is interefed in the following cases:

1. $m[\, \mathtt{in}\, n.\, P_1 \mid P_2 \,] \mid n[\, Q \,] \mid n[\, R \,]$
2. $m[\, \mathtt{in}\, n.\, P_1 \mid P_2 \,] \mid n[\, Q \,] \mid h[\, \mathtt{in}\, n.\, R_1 \mid R_2 \,]$
3. $\mathtt{open}\, m \mid m[\, \mathtt{in}\, n.\, P_1 \mid P_2 \,] \mid n[\, Q \,]$
4. $h[\, m[\, \mathtt{in}\, n.\, P_1 \mid \mathtt{out}\, h.\, P_3 \mid P_2 \,] \mid n[\, Q \,]\,]$ (with h possibly equal to n)
5. $m[\, \mathtt{in}\, n.\, P_1 \mid \mathtt{in}\, h.\, P_3 \mid P_2 \,] \mid n[\, Q \,] \mid h[\, R \,]$
6. $\mathtt{open}\, n \mid m[\, \mathtt{in}\, n.\, P_1 \mid P_2 \,] \mid n[\, Q \,]$
7. $m[\, \mathtt{in}\, n.\, P_1 \mid P_2 \,] \mid n[\, \mathtt{in}\, h.\, Q_1 \mid Q_2 \,] \mid h[\, R \,]$ (with h possibly equal to n)
8. $h[\, m[\, \mathtt{in}\, n.\, P_1 \mid P_2 \,] \mid n[\, \mathtt{out}\, h.\, Q_1 \mid Q_2 \,] \mid R \,]$

where all but the first two are **grave interferences**

Example: renaming

Cardelli-Gordon's encoding of renaming:

$$n \;\mathsf{be}\; m.\, P \;\triangleq\; m[\, \mathtt{out}\; n.\, \mathtt{open}\; n.\, P \,] \mid \mathtt{in}\; m$$

$$\begin{array}{rl}
 & n[\, n \;\mathsf{be}\; m.\, P \mid Q \,] = n[\, m[\, \mathtt{out}\; n.\, \mathtt{open}\; n.\, P \,] \mid \mathtt{in}\; m \mid Q \,] \\
\longrightarrow & m[\, \mathtt{open}\; n.\, P \,] \mid n[\, \mathtt{in}\; m \mid Q \,] \\
\longrightarrow & m[\, \mathtt{open}\; n.\, P \mid n[\, Q \,]\,] \\
\longrightarrow & m[\, P \mid Q \,]
\end{array}$$

$$
\begin{array}{rl}
& n[\, n\ \mathsf{be}\ m .\ P \mid Q\,] = n[\, m[\, \mathtt{out}\ n .\ \mathtt{open}\ n .\ P\,] \mid \mathtt{in}\ m \mid Q\,] \\
\longrightarrow & m[\, \mathtt{open}\ n .\ P\,] \mid n[\, \mathtt{in}\ m \mid Q\,] \\
\longrightarrow & m[\, \mathtt{open}\ n .\ P \mid n[\, Q\,]\,] \\
\longrightarrow & m[\, P \mid Q\,]
\end{array}
$$

From [CG]:

“This operation is not atomic: a movement initiated by Q may disrupt it. ... one can add a lock within n to synchronise renaming with any movement by Q.”

$$
\begin{aligned}
& n[\, n\ \mathtt{be}\ m.\, P \mid Q \,] = n[\, m[\, \mathtt{out}\ n.\, \mathtt{open}\ n.\, P \,] \mid \mathtt{in}\ m \mid Q \,] \\
\longrightarrow\quad & m[\, \mathtt{open}\ n.\, P \,] \mid n[\, \mathtt{in}\ m \mid Q \,] \\
\longrightarrow\quad & m[\, \mathtt{open}\ n.\, P \mid n[\, Q \,] \,] \\
\longrightarrow\quad & m[\, P \mid Q \,]
\end{aligned}
$$

Other ways in which the protocol may be corrupted:

- the environment opens n or m;
- the environment contains an ambient named m;
- the environment contains an ambient named n that tries to enter m;
- $Q = \mathtt{open}\ k$ and the the environment contains ambients $k[\, \mathtt{in}\ n.\, \mathtt{in}\ h \,]$ and $h[\, R \,]$

Furthermore, "synchronising any movement by Q" may be heavy:

(One would probably have to stop **all** activities of Q. Example: even $Q = k[\, Q' \,]$ may not be OK, for $Q' \triangleq \mathtt{out}\ n.\, \mathtt{open}\ n$.)

CONTENTS

Mobile Safe Ambients (SA)

[Sangiorgi, Levi, 00]

In SA, grave interferences are avoided by means of

- a modification of the **syntax** (addition of **coactions** $\overline{\mathtt{in}}\,n, \overline{\mathtt{out}}\,n, \overline{\mathtt{open}}\,n$)

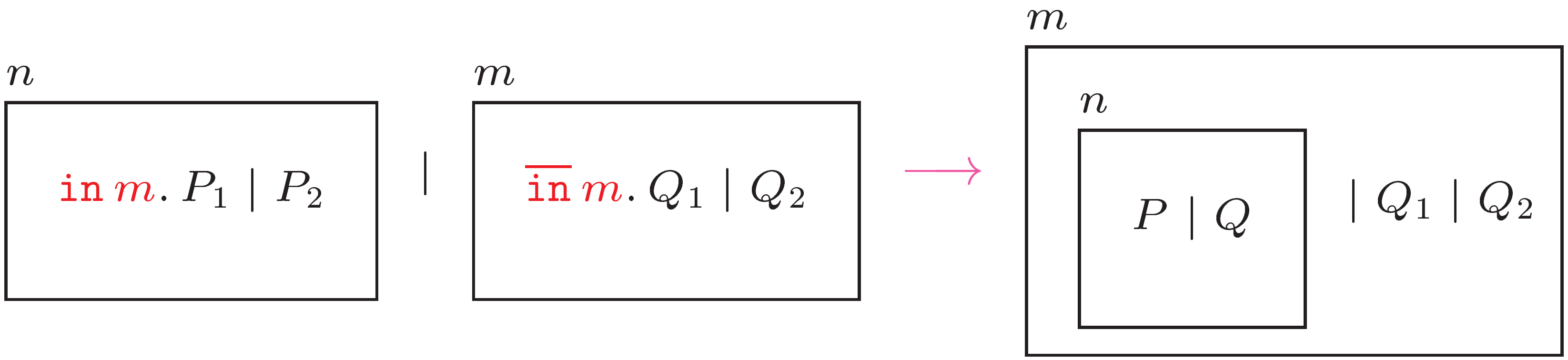
  [to make sure that any movement takes place only if *both* participants agree.]

- a **type system** (**single-threaded** and **immobility** types)

  [to control the movement of ambients]

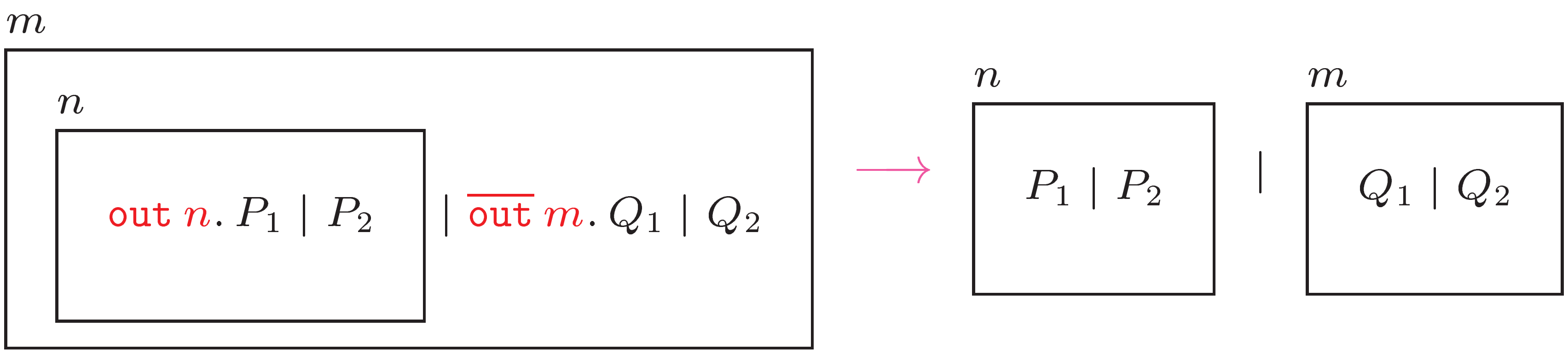
Also, coactions allow us to control **when** an ambient is opened or crossed

The `in` movement in SA



$$n[\, \mathtt{in}\, m.\, P_1 \mid P_2 \,] \mid m[\, \overline{\mathtt{in}}\, m.\, Q_1 \mid Q_2 \,] \longrightarrow m[\, n[\, P_1 \mid P_2 \,] \mid Q_1 \mid Q_2 \,]$$

The `out` movement in SA



$$m[\, n[\, \mathtt{out}\ m.\ P_1 \mid P_2 \,] \mid \overline{\mathtt{out}}\ m.\ Q_1 \mid Q_2 \,] \longrightarrow n[\, P_1 \mid P_2 \,] \mid m[\, Q_1 \mid Q_2 \,]$$

The open movement in SA

$$\mathtt{open}\, n.\, P \mid \overset{n}{\boxed{\overline{\mathtt{open}}\, n.\, Q_1 \mid Q_2}} \longrightarrow P \mid Q_1 \mid Q_2$$

$$\mathtt{open}\, n.\, P \mid n[\, \overline{\mathtt{open}}\, n.\, Q_1 \mid Q_2 \,] \longrightarrow P \mid Q_1 \mid Q_2$$

CONTENTS

Encoding of MA into SA

$$\{\![\, n[\, P\,]\,]\!\} \triangleq n[\, !\overline{\mathtt{in}}\, n \mid !\overline{\mathtt{out}}\, n \mid !\overline{\mathtt{open}}\, n \mid \{\![\, P\,]\!\}\,]$$

Typed SA

Two main forms of types for ambients

Single-threaded ambients	$\sim$	travelling agents
Immobile ambients	$\sim$	phisical sites

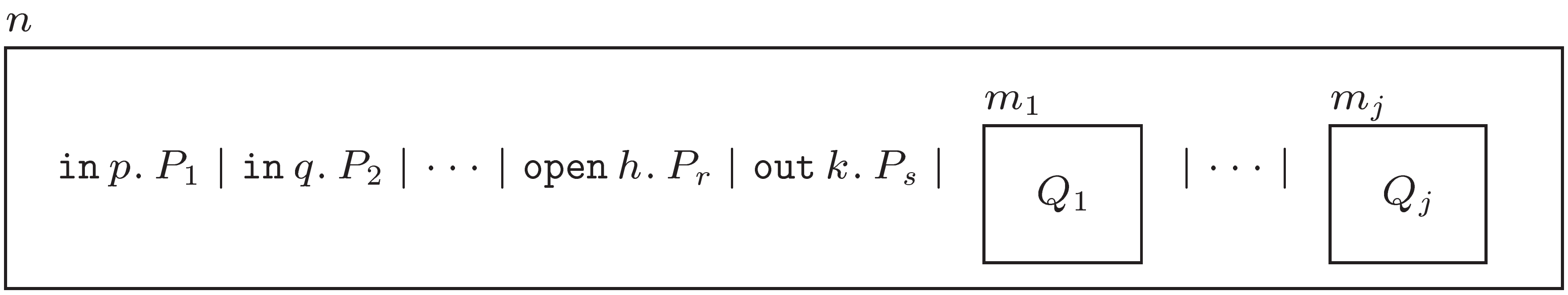
Theorem [no grave interferences]

Suppose P is well-typed in a type environment Γ. Suppose also that the type of each name defined in Γ and each restricted name in P is immobile or ST. For all P' such that $P \Longrightarrow P'$, no grave interferences occur in P'.

Single-threadness types

The capabilities of the local processes of an ambient control the activities of that ambient.

In standard Ambients: **several** local processes can exercise a capability at the same time.



(the local processes are the 'pilots')

In **Single-threaded (ST) ambient**: at any moment at most **one** process has the control thread.

Inside an ST ambient:

- the control thread need not be on the same process for ever:
  the thread can move by mean of an `open` movement or a communication
- several activities may take place concurrently

Non examples:

$$n[\, \mathtt{open}\, m \mid \mathtt{in}\, h \mid m[\, Q \,] \,] \qquad\qquad n[\, \mathtt{in}\, h \mid \mathtt{in}\, h \mid m[\, Q \,] \,]$$

Examples:

$$n[\, \mathtt{in}\, h.\, \overline{\mathtt{out}}\, m \mid p[\, \mathtt{in}\, m.\, Q \,] \mid m[\, R \,] \,] \qquad\qquad n[\, \mathtt{open}\, m.\, h[\, R \,] \mid m[\, \overline{\mathtt{open}}\, m.\, \mathtt{in}\, k \,] \,]$$

- To gain flexibility, we also allow messages carrying the thread. The control thread may then move also as a result of the consumption of a message:

$$n[\,\mathtt{in}\,h.\,\langle M\rangle \mid (x).\,x.\,Q\,] \mid h[\,\overline{\mathtt{in}}\,h.\,R\,] \quad \longrightarrow \; \longrightarrow h[\,R \mid n[\,M.\,Q\,]\,]$$

Immobility types

Immobile ambients are ambients that:

- cannot jump into or out of other ambients;
- cannot be opened.

Other ambients can jump into an immobile ambient, but cannot be opened there.

An immobile ambient can however be inside another one that jumps.

Non examples:

$$n[\, \mathtt{in}\, h.\, P \mid Q \,] \qquad\qquad n[\, \overline{\mathtt{open}}\, n.\, P \mid Q \,]$$

Examples:

$$n[\, !\overline{\mathtt{in}}\, n \mid !\overline{\mathtt{out}}\, n \mid h[\, \mathtt{out}\, n.\, Q \,]\,]$$

$$n[\, \overline{\mathtt{in}}\, n \mid \mathtt{open}\, h \mid (x).\, P \,] \mid h[\, \mathtt{in}\, n.\, \overline{\mathtt{open}}\, h.\, \langle M \rangle \,]$$

CONTENTS

Algebraic laws: examples

$$n[\,\overline{\texttt{out}}\,n.\,P \mid m[\,\texttt{out}\,n.\,Q\,]\,] \quad\approx\quad m[\,Q\,] \mid n[\,P\,]$$
$$k[\,\texttt{open}\,m.\,P \mid m[\,\overline{\texttt{open}}\,m.\,Q\,]\,] \quad\approx\quad k[\,P \mid Q\,]$$

Suppose ambients n and m are ST:

$$\boldsymbol{\nu} n\,(m[\,\texttt{in}\,n.\,P_1 \mid P_2\,] \mid n[\,\overline{\texttt{in}}\,n.\,Q_1 \mid Q_2\,]) \approx \boldsymbol{\nu} n\,(n[\,Q_1 \mid Q_2 \mid m[\,P_1 \mid P_2\,]\,])$$

If, in addition, $\texttt{in}\,n$, $\overline{\texttt{in}}\,n$ are *linear*, then

$$m[\,\texttt{in}\,n.\,P_1 \mid P_2\,] \mid n[\,\overline{\texttt{in}}\,n.\,Q_1 \mid Q_2\,] \approx n[\,Q_1 \mid Q_2 \mid m[\,P_1 \mid P_2\,]\,]$$

Example: renaming in SA

$n\,\mathtt{be}\,m.\,P \triangleq m[\,\mathtt{out}\,n.\,\overline{\mathtt{in}}\,m.\,\mathtt{open}\,n.\,P\,] \mid \overline{\mathtt{out}}\,n.\,\mathtt{in}\,m.\,\overline{\mathtt{open}}\,n$ (which respect ST for n, m)

$n[\,\boldsymbol{\nu} m\,(n\,\mathtt{be}\,m.\,P) \mid Q\,] \approx \boldsymbol{\nu} m\,(m[\,P \mid Q\,])$ (thus the **target** name m is only known to P)

In SA we can define a more robust renaming protocol:

$n\,\mathtt{be}^{+}\,m.\,P \triangleq \boldsymbol{\nu} h\,(\;h[\,\mathtt{out}\,n.\,\overline{\mathtt{in}}\,h.\,\overline{\mathtt{out}}\,h \mid m[\,\overline{\mathtt{in}}\,m.\,\mathtt{open}\,n\,]\,] \mid \overline{\mathtt{out}}\,n.\,\mathtt{in}\,h.\,\mathtt{in}\,m.\,\overline{\mathtt{open}}\,n.\,\mathtt{out}\,h.\,P)$

where h is any fresh name. We have, if n is ST:

$$\boldsymbol{\nu} n\,(n[\,n\,\mathtt{be}^{+}\,m.\,P \mid Q\,]) \approx \boldsymbol{\nu} n\,(m[\,P \mid Q\,])$$

Encoding π-calculus (ie, channels) into SA

Some problems in the previous encoding of channels in Ambients:

- encoding non-compositional
- each channel needs a centralised channel ambient
- channel ambients are immobile, but the type systems for Ambients are too weak to recognize this.
- correctness proof hard

Encoding channels into SA

$$
\begin{array}{rcl}
\{\![\, p(q_1..q_n).\, P \,]\!\} & \triangleq & p[\, \overline{\mathtt{in}}\, p.\, \mathtt{open}\, p.\, (q_1..q_n).\, \overline{\mathtt{open}}\, p.\, \{\![\, P \,]\!\} \,] \\
 & & \mid \mathtt{open}\, p \\
\{\![\, \overline{p}\langle q_1..q_n\rangle \,]\!\} & \triangleq & p[\, \mathtt{in}\, p.\, \overline{\mathtt{open}}\, p.\, \langle q_1..q_n\rangle \,] \\
\{\![\, \boldsymbol{\nu} p\; P \,]\!\} & \triangleq & \boldsymbol{\nu} p\; \{\![\, P \,]\!\} \\
\{\![\, P \mid Q \,]\!\} & \triangleq & \{\![\, P \,]\!\} \mid \{\![\, Q \,]\!\} \\
\{\![\, !P \,]\!\} & \triangleq & !\{\![\, P \,]\!\} \\
\{\![\, \mathbf{0} \,]\!\} & \triangleq & \mathbf{0}
\end{array}
$$

$$
\begin{array}{rcl}
\{\![\, p(q_1..q_n).\, P \,]\!\} & \triangleq & \boldsymbol{\nu} k \\
 & & (p[\, \overline{\mathtt{in}}\, p.\, \mathtt{open}\, p.\, (q_1..q_n).\, (k[\, \mathtt{out}\, p.\, \overline{\mathtt{open}}\, k.\, \{\![\, P \,]\!\} \,] \mid \overline{\mathtt{out}}\, p) \,] \\
 & & \mid \mathtt{open}\, k)
\end{array}
$$

$$\overline{p}\langle r_1 \,.\,.\, r_n\rangle \mid p(q_1 \,.\,.\, q_n).\, P \longrightarrow P\{\widetilde{r}/\widetilde{q}\}$$

$$\begin{array}{ll}
& p[\, \mathtt{in}\, p.\, \overline{\mathtt{open}}\, p.\, \langle r_1 \,.\,.\, r_n\rangle \,] \quad \mid p[\, \overline{\mathtt{in}}\, p.\, \mathtt{open}\, p.\, (q_1 \,.\,.\, q_n).\, \overline{\mathtt{open}}\, p.\, \{\![\, P \,]\!\} \,] \\
& \qquad\qquad\qquad\qquad\quad \mid \mathtt{open}\, p \\
\longrightarrow & p[\, p[\, \overline{\mathtt{open}}\, p.\, \langle r_1 \,.\,.\, r_n\rangle \,] \mid \mathtt{open}\, p.\, (q_1 \,.\,.\, q_n).\, \overline{\mathtt{open}}\, p.\, \{\![\, P \,]\!\} \,] \\
& \mid \mathtt{open}\, p \\
\longrightarrow & p[\, \langle r_1 \,.\,.\, r_n\rangle \mid (q_1 \,.\,.\, q_n).\, \overline{\mathtt{open}}\, p.\, \{\![\, P \,]\!\} \,] \\
& \mid \mathtt{open}\, p \\
\longrightarrow & p[\, \overline{\mathtt{open}}\, p.\, \{\![\, P\{\widetilde{r}/\widetilde{q}\} \,]\!\} \,] \\
& \mid \mathtt{open}\, p \\
\longrightarrow & \{\![\, P\{\widetilde{r}/\widetilde{q}\} \,]\!\}
\end{array}$$

Distributed implementation of SA

In **standard Ambients**:

Each movement operation involves ambients on different hierarchical levels, and may involve several ambients at of same level.

In **SA**:

Multiple synchronisations among all ambients affected by a movement are not necessary, due to the absence of grave interferences **[Sangiorgi, Valente 2001]**

[if the local processes of two ambients agree on a movements, the other local processes cannot prevent it]

Some other Ambient-like calculi

- Boxed Ambients, BA **[Bugliesi, Castagna, Crafa]**
  - direct communications between parents and child; no `open`
- NBA **[Bugliesi, Crafa, Merro, Sassone]**
  - SA + Boxed Ambients (removing grave interferences from BA)
- Robust Ambients **[Guan, Yang, You]**
  - modify parameters in coactions of SA
- Channeled Ambients **[Pericaas, Geertsen]**
  - channel communications inside ambients