Mobile Computing

Mobile Ambients II

Pascal Zimmer

pzimmer@daimi.au.dk

BRICS

Mobile Ambients – Syntax

P, Q	$::=$	$(\nu n)P$	restriction
		$\mathbf{0}$	inactivity
		$P \mid Q$	composition
		$!P$	replication
		$M[P]$	ambient
		$M.P$	action
		$(x_1, \ldots, x_k).P$	input
		$\langle M_1, \ldots, M_k \rangle$	async output
M	$::=$	$n \mid x \mid in\ M \mid out\ M \mid open\ M$	expressions
		$\mid M_1.M_2 \mid \varepsilon$	

Mobile Ambients – Semantics

Entering and exiting an ambient:

$$n[\, in\, m\, .\, P \mid Q] \mid m[R] \;\rightarrow\; m[n[P \mid Q] \mid R]$$
$$m[n[\, out\, m\, .P \mid Q] \mid R] \;\rightarrow\; n[P \mid Q] \mid m[R]$$

Opening an ambient:

$$open\, n\, .P \mid n[Q] \;\rightarrow\; P \mid Q$$

Asynchronous communication:

$$(x_1, \ldots, x_k).P \mid \langle M_1, \ldots, M_k \rangle \;\rightarrow\; P\{^{M_1}/_{x_1}, \ldots, ^{M_k}/_{x_k}\}$$

Implementation

- A monothread implementation is easy to write...

Implementation

- A monothread implementation is easy to write...
- ... but not very efficient...

Implementation

- A monothread implementation is easy to write...
- ... but not very efficient...
- ... and much more difficult to write in a distributed setting !

Implementation

- A monothread implementation is easy to write...
- ... but not very efficient...
- ... and much more difficult to write in a distributed setting !

$\Rightarrow$ A distributed abstract machine for safe ambients (Sangiorgi, Valente 2001) (for well-typed monothreaded safe ambients)

Equational theory

References

- A. D. Gordon and L. Cardelli, *Equational properties of mobile ambients*, FoSSaCS'99 (and MSCS)
- M. Merro and M. Hennessy, *Bisimulation congruences in safe ambients*, POPL'02
- M. Merro and F. Zappa Nardelli, *Bisimulation proof methods for mobile ambients*, ICALP'03

Another notion of barb (CG)

- Exhibition of a name:

$$P \downarrow n \ \triangleq\ P \equiv (\nu \vec{m})(n[P'] \mid P'')$$

with $n \notin \vec{m}$

Another notion of barb (CG)

- Exhibition of a name:

$$P \downarrow n \ \triangleq\ P \equiv (\nu \vec{m})(n[P'] \mid P'')$$

  with $n \notin \vec{m}$

- Convergence to a name:

$$P \Downarrow n \ \triangleq\ P \longrightarrow^{*} \downarrow n$$

Contextual equivalence (CG)

- Contextual equivalence:

$$P \simeq Q \triangleq C[P] \Downarrow n \Leftrightarrow C[Q] \Downarrow n$$

  for any name n and context C such that $C[P]$ and $C[Q]$ are closed.

Contextual equivalence (CG)

- Contextual equivalence:

$$P \simeq Q \triangleq C[P] \Downarrow n \Leftrightarrow C[Q] \Downarrow n$$

  for any name n and context C such that $C[P]$ and $C[Q]$ are closed.

- $\simeq$ is a congruence and contains $\equiv$
- Proof technique: labelled transition system... much tougher than in π !

Examples

- Opening:

$$(\nu n)(n[] \mid open\, n.P) \simeq P \quad \text{if } n \notin fn(P)$$

Examples

- Opening:

$$(\nu n)(n[] \mid open\, n.P) \simeq P \quad \text{if } n \notin fn(P)$$

- Perfect firewall:

$$(\nu n)n[P] \simeq \mathbf{0} \quad \text{if } n \notin fn(P)$$

Examples

- Opening:

$$(\nu n)(n[] \mid open\, n.P) \simeq P \quad \text{if } n \notin fn(P)$$

- Perfect firewall:

$$(\nu n)n[P] \simeq \mathbf{0} \quad \text{if } n \notin fn(P)$$

- Firewall and agent:

$$(\nu k\; k'\; k'')(Agent \mid Firewall) \simeq (\nu w)w[Q \mid P]$$

Reduction barbed congruence (MZN)

- A slightly modified ambient calculus (systems vs processes, replication of actions)...

Reduction barbed congruence (MZN)

- A slightly modified ambient calculus (systems vs processes, replication of actions)...
- **Reduction barbed congruence**:
  The largest symmetric relation over systems which is reduction closed (weakly), contextual and barb preserving (weakly).

Reduction barbed congruence (MZN)

- A slightly modified ambient calculus (systems vs processes, replication of actions)...
- **Reduction barbed congruence**:
  The largest symmetric relation over systems which is reduction closed (weakly), contextual and barb preserving (weakly).
- A labelled transition system...

Reduction barbed congruence (MZN)

- A slightly modified ambient calculus (systems vs processes, replication of actions)...
- **Reduction barbed congruence**:
  The largest symmetric relation over systems which is reduction closed (weakly), contextual and barb preserving (weakly).
- A labelled transition system...
- A definition of bisimilarity...

Reduction barbed congruence (MZN)

- A slightly modified ambient calculus (systems vs processes, replication of actions)...
- **Reduction barbed congruence**:
  The largest symmetric relation over systems which is reduction closed (weakly), contextual and barb preserving (weakly).
- A labelled transition system...
- A definition of bisimilarity...
- Bisimilarity and barbed congruence coincide !

Expressivity

Motivations

- Theoretical interest: What makes the ambient calculus so expressive ? What are the minimal constructs ?
- To simplify future works by decreasing the number of cases to study.
- Find ideas and strategies for an implementation.

Some expressivity results

$!$ **vs** rec

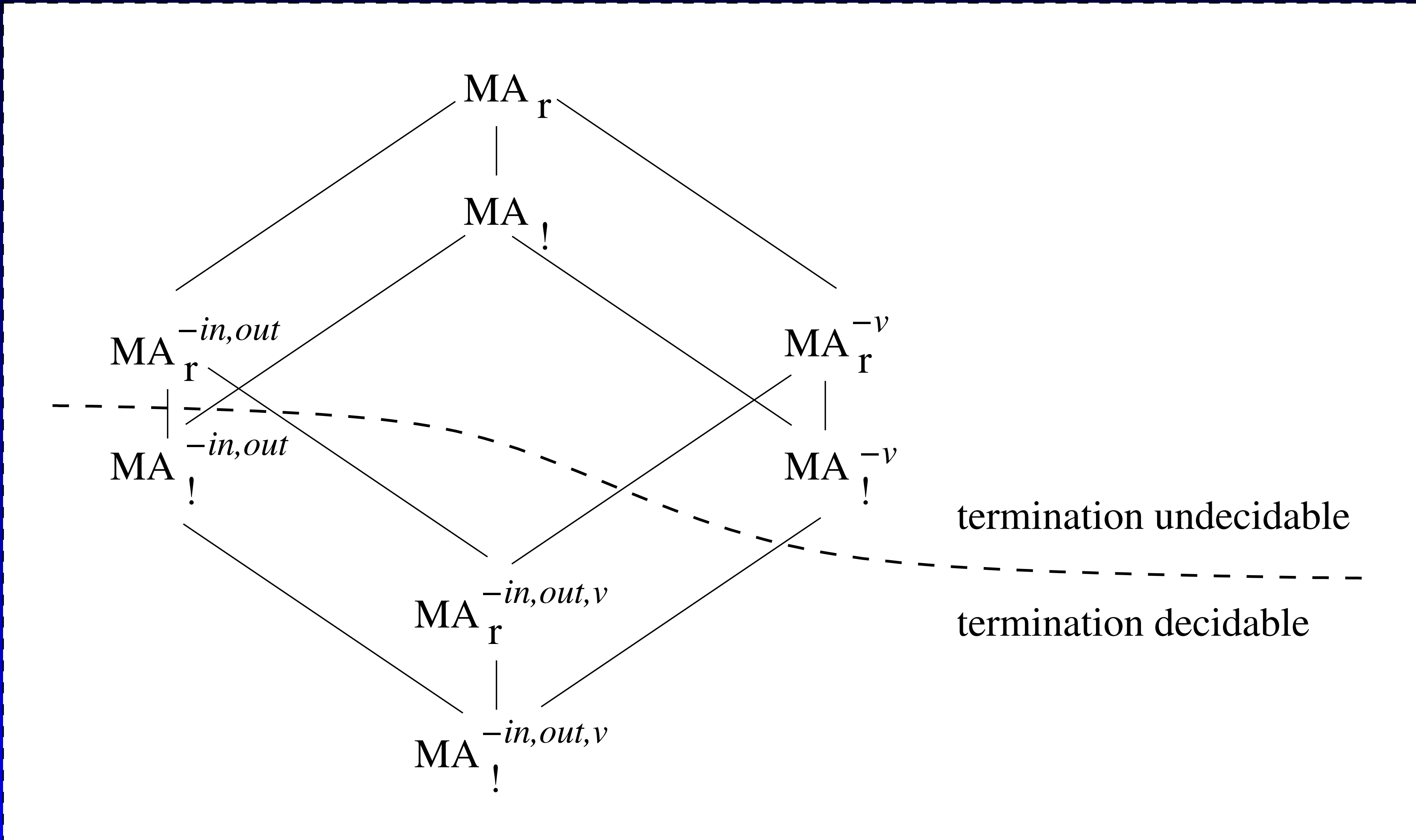
- $!P$ can always be encoded as $recX.(P \mid X)$
- No converse encoding is known
- rec is probably "more" expressive than $!$

Some expressivity results

Busi, Zavattaro 2002

- Instead of looking at Turing machines, they consider the decidability of termination
- For a calculus with ! or rec
- With or without movements (in and out)
- With or without restriction (ν)
- Proofs are based on an encoding of RAMs and a reduction to well-structured transition systems

Some expressivity results



Some expressivity results

Boneva, Talbot 2003

- Consider the calculus (with replication) without $open$
- Reachability problem (given P, Q, does $P \longrightarrow^* Q$ hold ?) is undecidable...
- ... but becomes decidable if we replace

$$!P \equiv P \mid !P$$

with an oriented reduction rule:

$$!P \;\rightarrow\; P \mid !P$$

Some expressivity results

Boneva, Talbot 2003

- Name-convergence problem (given P, n, does $P \Downarrow n$ hold ?) is undecidable for both versions
- Model-checking problem (against ambient logics) is undecidable for both versions

Expressiveness of Pure Ambients

- It has been shown (Cardelli and Gordon) that pure ambients are *Turing-powerful*.

Expressiveness of Pure Ambients

- It has been shown (Cardelli and Gordon) that pure ambients are *Turing-powerful*.
- We show that the *Synchronous* π*-Calculus* can be encoded into pure ambients in a “satisfactory”way.

Expressiveness of Pure Ambients

- It has been shown (Cardelli and Gordon) that pure ambients are *Turing-powerful*.
- We show that the *Synchronous* π*-Calculus* can be encoded into pure ambients in a "satisfactory"way.
- Turing machines are not a satisfactory reference in a distributed and concurrent world.

$\Rightarrow$ better refer to π

Expressiveness of Pure Ambients

- It has been shown (Cardelli and Gordon) that pure ambients are *Turing-powerful*.
- We show that the *Synchronous* π*-Calculus* can be encoded into pure ambients in a "satisfactory"way.
- Turing machines are not a satisfactory reference in a distributed and concurrent world.

$\Rightarrow$ better refer to π

- We could encode π into pure ambients through Turing machines.
- However, the encoding would not be *compositional*.

Pure Safe Ambients

They are the safe ambients, without communication primitives and rules.
Syntax:

$$
\begin{array}{rcll}
P & ::= & (\nu n)\, P & \text{restriction} \\
 & \mid & \mathbf{0} & \text{nil process} \\
 & \mid & P \mid Q & \text{parallel composition} \\
 & \mid & !P & \text{replication} \\
 & \mid & n[P] & \text{ambient} \\
 & \mid & Cap.P & \text{capability}
\end{array}
$$

$$Cap ::= \; in\; n \mid \overline{in}\; n \mid out\; n \mid \overline{out}\; n \mid open\; n \mid \overline{open}\; n$$

Pure Safe Ambients

Entering and exiting an ambient:

$$n[\, in\, m\, .\, P \mid Q] \mid m[\, \overline{in}\, m\, .R \mid S] \hookrightarrow m[n[P \mid Q] \mid R \mid S]$$

$$m[n[\, out\, m\, .P \mid Q] \mid \overline{out}\, m\, .R \mid S] \hookrightarrow n[P \mid Q] \mid m[R \mid S]$$

Opening an ambient:

$$open\, n\, .P \mid n[\, \overline{open}\, n\, .Q] \hookrightarrow P \mid Q$$

Outline

- Introduction

  π-calculus $\longleftrightarrow$ π_{esc}-calculus $\longleftrightarrow$ pure ambients
- Definition of the π_{esc}-calculus and operational correspondence with the π-calculus
- Encoding the π_{esc}-calculus in pure ambients and operational correspondence
- Final encoding and main result
- Conclusion and future work

Reminder: Synchronous π

Syntax:

$$
\begin{array}{lclll}
P & ::= & (\nu n)\, P & \text{restriction} & M ::= n \in Name \\
 & | & \mathbf{0} & \text{nil process} & \quad\;\; | \; x \in Var \\
 & | & P \mid Q & \text{parallel composition} & \\
 & | & !P & \text{replication} & \\
 & | & \overline{M}\langle M' \rangle.P & \text{output} & \\
 & | & M(x).P & \text{input} &
\end{array}
$$

Communication rule:

$$\overline{n}\langle m \rangle.P \mid n(x).Q \longrightarrow P \mid Q\{^m/_x\}$$

π_{esc}-Calculus: Syntax

Same syntax as the π-calculus, adding:

P	$::=$	$\ldots$	
	$\mid$	$[n : S]$	explicit channel
	$\mid$	$(\nu x : M)\ P$	explicit variable ($x \neq M$)
S	$::=$	ε	empty channel
	$\mid$	$S \mid S'$	parallel composition
	$\mid$	$\langle M \rangle.P$	concretion
	$\mid$	$(x).P$	abstraction

π_{esc}: Operational Semantics

- Rules are of the form $\sigma : P \longmapsto P'$: a process P reduces to a process P' in the environment σ (= a substitution binding every free variable of P).
- Substituting a variable in a prefix by its value:

$$\frac{x\sigma = M}{\sigma : \overline{x}\langle M' \rangle.P \longmapsto \overline{M}\langle M' \rangle.P}$$

$$\frac{x\sigma = M}{\sigma : x(y).P \longmapsto M(y).P}$$

π_{esc}: Operational Semantics

- Output and input on a channel:

$$\frac{}{\sigma : [n : S] \mid \overline{n}\langle M\rangle.P \;\longmapsto\; [n : S \mid \langle M\rangle.P]}$$

$$\frac{}{\sigma : [n : S] \mid n(x).P \;\longmapsto\; [n : S \mid (x).P]}$$

π_{esc}: Operational Semantics

- Effective communication in a channel, creation of a new variable and activation of the continuations:

$$\frac{x \neq M}{\sigma : [n : S \mid \langle M \rangle.P \mid (x).Q] \longmapsto [n : S] \mid P \mid (\nu x : M)\, Q}$$

π_{esc}: Operational Semantics

- Effective communication in a channel, creation of a new variable and activation of the continuations:

$$\frac{x \neq M}{\sigma : [n : S \mid \langle M \rangle.P \mid (x).Q] \longmapsto [n : S] \mid P \mid (\nu x : M)\, Q}$$

- Integration of a variable in the environment:

$$\frac{x \notin dom(\sigma) \qquad \{^{M}/_{x}\} \uplus \sigma : P \longmapsto P'}{\sigma : (\nu x : M)\, P \longmapsto (\nu x : M)\, P'}$$

- Reduction under (νn) , in parallel or by structural congruence $\equiv$...

Valid Processes and Channel Closure

- Channels can be unreachable: $\overline{n}\langle m\rangle.[p : S]$ or too numerous: $[n : S] \mid [n : S'] \mid \overline{n}\langle m\rangle.P \mid n(x).Q$

  $\Rightarrow$ A simple type system to avoid those *invalid* processes. Validity is preserved by reduction.

Valid Processes and Channel Closure

- Channels can be unreachable: $\overline{n}\langle m\rangle.[p : S]$ or too numerous: $[n : S] \mid [n : S'] \mid \overline{n}\langle m\rangle.P \mid n(x).Q$
  $\Rightarrow$ A simple type system to avoid those *invalid* processes. Validity is preserved by reduction.
- Channels can be missing: $(\nu n)\ (\overline{n}\langle m\rangle.P \mid n(x).Q)$
  $\Rightarrow$ A *channel closure* (w.r.t. a substitution σ) $cl_\sigma(P)$ to add missing channels. P is *channel-closed* if all channels are present.

From π_{esc} to π

Translating a π_{esc}-process in a intuitively “equivalent” π-process:

$$
\begin{array}{ll}
[\![[n : S]]\!] \triangleq [\![S]\!]_n & [\![\varepsilon]\!]_n \triangleq \mathbf{0} \\
[\![(\nu x : M)\ P]\!] \triangleq [\![P]\!]\{^M/_x\} & [\![S \mid S']\!]_n \triangleq [\![S]\!]_n \mid [\![S']\!]_n \\
 & [\![\langle M \rangle .P]\!]_n \triangleq \overline{n}\langle M \rangle .[\![P]\!] \\
 & [\![(x).P]\!]_n \triangleq n(x).[\![P]\!]
\end{array}
$$

($[\![.]\!]$ is an homomorphism for all other constructs)

Operational Correspondence $\pi_{esc} \longrightarrow \pi$

Proposition 1 If $\emptyset : P \longmapsto Q$, then $[\![P]\!]\ \mathcal{R}\ [\![Q]\!]$, where $\mathcal{R}$ is either $\equiv$ or $\longrightarrow$.

π-calculus	$\Longleftarrow$	π_{esc}-calculus
$[\![P]\!]$		P
$\mathcal{R}$		$\downarrow$
$[\![Q]\!]$		Q

Operational Correspondence $\pi \longrightarrow \pi_{esc}$

Proposition 2 If a process P is channel-closed w.r.t. $\varnothing$, valid and without free variables, and if $[\![P]\!] \longrightarrow Q$, then there is a process P' such that $\varnothing : P \longmapsto^{+} P'$ and $[\![P']\!] \equiv Q$.

$$
\begin{array}{ccc}
\pi\text{-calculus} & \Longrightarrow & \pi_{esc}\text{-calculus} \\
\hline
[\![P]\!] & & P \\
 & & \downarrow \\
\downarrow & & \vdots \\
 & & \downarrow \\
Q \equiv [\![P']\!] & & P'
\end{array}
$$

Encoding π_{esc} into Pure Ambients

- Actors “communicate” by a request/server mechanism:
  - A server is a replicated process which tries to inject its code into requests and take their control.
  - A request is an ambient allowing the code injection and execution.
- A channel is simulated by an ambient n receiving and processing $read$ and $write$ requests.
- A variable is simulated by an ambient x receiving and processing $read$ and $write$ requests by forwarding them to M.

Operational Correspondence

Proposition 3 If $\sigma : P \longmapsto Q$, then
$\{\!\{\sigma, P\}\!\} \stackrel{pr}{\hookrightarrow} \stackrel{aux*}{\hookrightarrow} \{\!\{\sigma, Q\}\!\}$.

Proposition 4 If $\{\!\{\sigma, P\}\!\} \stackrel{pr}{\hookrightarrow} Q$, then there is a process P' such that $\sigma : P \longmapsto P'$ and $Q \stackrel{aux*}{\hookrightarrow} \{\!\{\sigma, P'\}\!\}$. Moreover, if $\sigma : P \longmapsto P''$ and $Q \stackrel{aux*}{\hookrightarrow} \{\!\{\sigma, P''\}\!\}$, then $P' \equiv P''$ (in other words P' is unique modulo $\equiv$).

Encoding π into Pure Ambients

- From the π-calculus to the π_{esc}-calculus: we only need to add channels, $(\nu n)\ P$ becomes $(\nu n)\ ([n : \varepsilon] \mid P)$.
- From the π_{esc}-calculus to pure ambients: P becomes $\{\!\{\varnothing, P\}\!\}$.
- The final encoding $\langle\!\langle P \rangle\!\rangle$ is the composition of the two previous encodings.
- It can be written directly, and not via the π_{esc}-calculus...

Main Result

Definition Let P be a π-process with no free variables and R a pure ambient process. We will say that P and R are equivalent (written $P \approx R$) if there is a π_{esc}-process Q such that Q is valid, channel-closed w.r.t. $\varnothing$, with no free variables, $P \equiv [\![Q]\!]$ and $\{\!\{\varnothing, Q\}\!\} \equiv R$.
It is routine to check that $P \approx \langle\!\langle P \rangle\!\rangle$ for every π-process P with no free variables.

Theorem Suppose $P \approx R$.

- If $P \longrightarrow P'$, then there is a process R' such that $R \hookrightarrow^{+} R'$ and $P' \approx R'$.
- If $R \stackrel{pr}{\hookrightarrow} R'$, then there is a process R'' such that $R' \stackrel{aux*}{\hookrightarrow} R''$, and either $P \approx R''$ or $P \longrightarrow P' \approx R''$.

Open Problems

- Proving a conjecture (with the help of an automatic demonstration tool) and state a stronger result for the operational correspondence
- Encoding the polyadic π-calculus (should be easy)
- Encoding the π-calculus in classical ambients instead of safe ambients (difficult ???)
- Main question: encoding ambients with communications into ambients without communications