On the Expressiveness of Pure Mobile Ambients

Pascal ZIMMER

INRIA Sophia Antipolis / ENS Lyon – FRANCE

August 21, 2000

Safe Ambients

The Ambient Calculus: A calculus designed to model mobility.

- An ambient is a bounded place in which computation happens.
- Ambients can be nested.
- An ambient can move into another one, out of another one or can be opened.

Safe Ambients: With additional coactions to prevent interferences.

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:

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

Asynchronous communication:

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

Pure Ambients

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

Syntax:

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

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

Expressiveness

- It has been shown (Cardelli and Gordon) that pure ambients are *Turing-complete*.
- We showed that the *Synchronous* π*-Calculus* can be encoded into pure ambients.

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.
- "Life-size" programming exercise in ambients.

Outline

- Introduction

$$\pi\text{-calculus} \longleftrightarrow \pi_{esc}\text{-calculus} \longleftrightarrow \text{pure ambients}$$

- 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

Background: the Synchronous π-Calculus

Syntax:

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

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

Other reduction rules: under (νn) , in parallel or by structural congruence $\equiv$.

π_{esc}-Calculus: Syntax

Same syntax as the π-calculus, adding:

$$
\begin{array}{rcll}
P & ::= & \ldots & \\
 & \mid & [n : S] & \text{explicit channel} \\
 & \mid & (\nu x : M)\ P & \text{explicit variable } (x \neq M) \\
 & & & \\
S & ::= & \varepsilon & \text{empty channel} \\
 & \mid & S \mid S' & \text{parallel composition} \\
 & \mid & \langle M \rangle.P & \text{concretion} \\
 & \mid & (x).P & \text{abstraction}
\end{array}
$$

π_{esc}-Calculus: 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:

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

- 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}-Calculus: 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.

- 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 the π_{esc}-Calculus to the π-Calculus

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

$$\llbracket [n : S] \rrbracket \triangleq \llbracket S \rrbracket_n \qquad \llbracket \varepsilon \rrbracket_n \triangleq \mathbf{0}$$

$$\llbracket (\nu x : M)\ P \rrbracket \triangleq \llbracket P \rrbracket \{^M/_x\} \qquad \llbracket S \mid S' \rrbracket_n \triangleq \llbracket S \rrbracket_n \mid \llbracket S' \rrbracket_n$$

$$\llbracket \langle M \rangle.P \rrbracket_n \triangleq \overline{n}\langle M \rangle.\llbracket P \rrbracket$$

$$\llbracket (x).P \rrbracket_n \triangleq n(x).\llbracket P \rrbracket$$

($\llbracket . \rrbracket$ is an homomorphism for all other constructs)

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

Proposition 1 If $\varnothing : P \longmapsto Q$, 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 $\pi \to \pi_{esc}$

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 the π_{esc}-Calculus in 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 $\{\!\{\sigma, P\}\!\} \overset{pr}{\hookrightarrow} \overset{aux*}{\hookrightarrow} \{\!\{\sigma, Q\}\!\}$.

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

Encoding the π-Calculus in 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... (which was the case at the beginning of our research)

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

Conclusion

- An encoding of the synchronous π-calculus in pure ambients
- A proof of its correction by operational correspondence results
- An interesting extension of the π-calculus (π_{esc})

Future Work

- 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