Higher-order, Mobility and Locations

Model-based Design of Distributed and Mobile Systems Lecture 10: Spring 2005

Thomas Hildebrandt
hilde@dtu.dk

Department of Theoretical Computer Science
IT University of Copenhagen

Overview of the Lecture

The structure of the lecture:
- Dichotomies in Calculi for Mobility
  - First vs. Higher-order
  - Implicit vs. Explicit Locations
  - Flat vs. Nested Locations
  - Code vs. Process Mobility (weak vs. strong)
  - Objective vs. Subjective (autonomous, agent) mobility
- Higher-order pi-calculus [Sangiorgi, 1992]
- The Local Area pi-Calculus [Chothia, 2002]
- Higher-order Mobile Embedded Resources (Homer) [Hildebrandt, Godskesen, Bundgaard, 2004]
- Mobile Ambients [Cardelli, 1998]
- Mandatory Exercise

Dichotomies in Models for Mobility

- **First-order vs. Higher-order:** Base-type objects (e.g., integers, names, ..) vs. any-type objects (e.g., processes) passed as messages
- **Implicit Locations:** Communication links determine the location
  - **Explicit Locations:** The location is explicit (and may determine communication links)
- **Flat vs. Nested Locations:** Locations cannot/can be included in other locations
- **Passive vs. Active Process Mobility:** Active (running) processes cannot/can move
- **Objective Mobility:** Entities are moved by external entities
  - **Subjective (agent) mobility:** Entities move by themselves
- **Linear vs. non-linear:** Mobile entities cannot/can be copied.

Higher-order \( \pi \)-calculus

The ideas of Bent Thomsen and Davide Sangiorgi

- We add higher-order input \( a(X).Q \) and output \( \pi(P).Q \) prefixes, and add process variables \( X \)
- (Passive) process-passing instead of name-passing

New rule for reaction

\[
\text{REACT} \quad \frac{a(X).P + P'}{ \langle \pi(Q).R + R' \rangle \rightarrow P|Q/X | R}
\]

Examples:
- \( \pi(R).P | a(X).X \rightarrow P | R \)
- \( \pi(R).P | a(X).0 \rightarrow P \)
Copying processes

Since an agent variable can occur several times in an agent, we can now copy agents
\[ a(X).(X | X) \mid \pi(Q).R \rightarrow Q | Q | R \]

So we can encode recursion and replication (note that it takes one reaction to unfold another copy).

- Encoding of replication
  \[ [\pi P] = (\nu a)(D | \pi(P | D)) \]
  \[ D = a(X).(X | \pi(X)), \text{ where } a \notin fn(P) \]

- Why do we have to send \( D \) with \( P \)?

Higher-order into First-order

- First-order \( \pi \)-calculus is expressive enough to encode higher-order \( \pi \)-calculus. The main theme of Sangiorgi’s Thesis “Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms”
- Principle of encoding: Pass a reference (and use triggers)
- Assume fresh name \( x \) for each agent variable \( X \)

\[
\begin{align*}
[\pi(P).Q] &= (\nu P)[\pi(P)].([Q] | [P].[P]), \text{ where } p \notin fn(P, Q) \\
[a(X).P] &= a(x).[P] \\
[X] &= \pi
\end{align*}
\]

Example

In HO\( \pi \):
\( \pi(R).P \mid a(X).(X | X) \rightarrow P | R | R \)

In \( \pi \)-calculus:
\begin{align*}
(\nu r)\pi(r).([P] | [r].[R]) \mid a(x)(\pi | X) &\rightarrow \\
(\nu r)\pi([P] | [r].[R] | [\pi | X]) &\rightarrow \\
(\nu r)\pi([P] | [r].[R] | [\pi | X]) &\rightarrow \sim [P] | [R] | [R]
\end{align*}

What do we require of an encoding?

Sangiorgi, p. 8:
1. Formal definition of the semantics of the two languages
2. Definition of the encoding from the source to the target language
3. Proof of the correctness of the encoding w.r.t. the semantics given

- 2. must be compositional: e.g. \( [P | Q] = [P] | [Q] \)
- 1. and 3. often based on operational semantics, comparing the behaviours. Requires a uniform definition of equivalence
- Often required to be fully abstract: \( P \sim Q \) if and only if \( [P] \sim [Q] \)

Presheaf models are a promising approach to denotational semantics for concurrency with a uniform definition of equivalence [Winskel, Cattani, Hildebrandt, Nygaard, ...].

Summary on Higher-order \( \pi \)

- First-order \( \pi \)-calculus: Linear, subjective, active process mobility between implicit, nested (overlapping) locations
- Higher-order \( \pi \)-calculus: Adds non-linear, objective, passive process passing
- Can be encoded in first-order \( \pi \)-calculus
Local Area $\pi$-calculus (la$\pi$) [Chothia, PhD thesis]

Recall the internet daemon (e.g. xinetd) from lecture 8:

Client:

$$(vc)\text{server}(finger,e) | c(x).\text{print}(x)$$

← ask server

$$c(x).\text{print}(x)$$

← print response

Server:

$$\text{server}(service, reply) . service(reply)$$

← inet daemon

$$\text{finger}(reply) . reply(user)$$

← finger daemon

$$\text{time}(reply) . reply(now)$$

← time daemon

Nothing stops the client from connecting directly to the finger service at the server

- la$\pi$ adds explicit, nested areas (locations)
- Binding of name depends on location and operation level of name

Local Area $\pi$ encoded in (polyadic) $\pi$

- Again, the standard first-order (polyadic) $\pi$-calculus is expressive enough to encode higher-order $\pi$-calculus. (EXPRESS’01 paper by Chothia and Stark)
- Principle of encoding: Replace local area communication with communication on a new channel representing an ether.
- An output action $\alpha(a,b)$ becomes $\alpha(a,e) \quad e$ is a name corresponding to the local area on which $a$ operates
- Encoding of input: Fetch input from ether and "match or resend"

$$\llbracket \alpha(a,b).P \rrbracket = \text{rec}X e(x,b). \text{if} \ x = a \ \llbracket P \rrbracket \text{else} \ \llbracket \alpha(x,b) \ | \ X \rrbracket$$

Can we encode la$\pi$ into monadic $\pi$?
Can the encodings of Higher-order and Local Area be combined to an encoding of HOla$\pi$ into $\pi$?

Summary on Local Area $\pi$

- First-order pi-calculus: Linear, subjective, active process mobility between implicit, nested (overlapping) locations
- Local Area pi-calculus: Adds explicit nested locations (areas) and
- Can be encoded in (polyadic) first-order pi-calculus
- May be possible to combine with HO encoding, but not obvious

Local Area $\pi$-calculus

We can now add areas (locations) to the internet daemon model:

Client:

$$Hilde = \text{host}[(vc)\text{server}(finger,e) | c(x).\text{print}(x)]$$

← ask server

$$c(x).\text{print}(x)$$

← print response

Server:

$$ITU = \text{host} [\text{server}(service, reply) . service(reply)]$$

← inet daemon

$$\text{finger}(reply) . reply(user)$$

← finger daemon

$$\text{time}(reply) . reply(now)$$

← time daemon

System:

$$\text{net}[Hilde \ | \ ITU]$$

Names:

- server and $c$ operate at net level;
- finger, time and print operate at host level;
Higher-order Mobile Embedded Resources

Hildebrandt, Godskesen, and Bundgaard (EXPRESS 2004 and ITU TR):
- No name-passing
- Higher-order input \( a(X).Q \) and output \( \pi(P).Q \) prefixes, and process variables \( X \) as in CHOCS and HOπ
- Higher-order take \( a(X).Q \) and (reactive) resource \( a(P).Q \) prefixes
- Nested names e.g. \( a.b \) and new rules for reaction, e.g.
  \[
  a.b(X).P \mid (a(b(Q).Q' \mid Q'')) \longrightarrow P\{Q/X\} \mid a(Q' \mid Q'')
  \]
  \[
  a.b(R).P \mid (a(b(X).Q \mid Q')) \longrightarrow P \mid a(Q\{R/X\} \mid Q')
  \]
- Full Homer also adds active resource \( a[P].Q \) prefixes and the rule
  \[
  P \longrightarrow P'
  \]
  \[
  a[P].Q \longrightarrow a[P'].Q
  \]

\( \pi \) and spi into Homer

- First-order \( \pi \)-calculus can be encoded into Homer, *EXPRESS’04 paper by Bundgaard, Hildebrandt, Godskesen*
- Principle of encoding: Encode names as resources that can perform send, receive and binding
- spi-calculus encryption/decryption can be encoded into Homer
- Principle of encoding: Encode encryption by nesting.

Summary on Homer

- No name-passing (can be encoded)
- Higher-order, non-linear, subjective passive and (re)active process mobility
- Explicit nested locations and nested names

Mobile Ambients

L. Cardelli and A. Gordon, Journal of TCS 2000:
- Subjective, linear mobility between explicit nested locations (ambients)
  \[
  a[in b.P \mid P'] \mid b(Q) \longrightarrow b[a[P \mid P'] \mid Q]
  \]
  \[
  a[out a.P \mid P'] \mid Q \longrightarrow b[P \mid P'] \mid a[Q]
  \]
  \[
  open b.P \mid b(Q) \longrightarrow P \mid Q
  \]
- Ambients are active (as in Homer):
  \[
  P \longrightarrow P'
  \]
  \[
  a[P] \longrightarrow a[P']
  \]
Ambient Examples

- **Locks:**
  
  - acquire $n.P \equiv \text{open } n.P$
  
  - release $n.P \equiv n[] \mid P$

- **Firewall:** $(a)a[P]$

- **CCS interaction, Renaming, Turing Machines...**

Mandatory Exercise

1. Apply the encoding of Higher-order $\pi$-calculus into first-order $\pi$-calculus to the following expression:

   $$\pi(c).c() \mid a(X).(X \mid b(n).\pi())$$

2. Infer reductions in the encoding corresponding to:

   $$\pi(b().c() \mid a(X).(X \mid b(n).\pi()) \rightarrow c() \mid b(c) \mid b(n).\pi()) \rightarrow c() \mid \pi()) \rightarrow 0$$

3. Consider a Local Area Higher-order $\pi$:

   $$\text{net}[\pi(b().c() \mid a(X).(X \mid b(n).\pi())]$$

   $$\rightarrow \text{net}[\pi(c()) \mid \text{host}[b]()]$$

   $$\rightarrow \text{net}[\pi(c()) \mid \text{host}()] \rightarrow \text{net}[\text{host}[0] \mid \text{host}[0]]$$

   and assume that $a$ and $c$ operate at net level and $b$ operates at host level. Explain what happens if we apply the encoding of 1. to the process in 3.

Exercise — Higher-order Encoding of Replication

Try to infer reactions of the higher-order encoding of replication.

- Encoding of replication

  $$[!P] = (\nu a)(D \mid \pi(P \mid D))$$

  $$D = a(X).(X \mid \pi(X)),$$ where $a \notin \text{fn}(P)$

- Why do we have to send $D$ with $P$?

All Done

If you have done all the exercises, come ask me for more ;)

Mandatory Exercise

1. Apply the encoding of Higher-order $\pi$-calculus into first-order $\pi$-calculus to the following expression:

   $$\pi(c).c() \mid a(X).(X \mid b(n).\pi())$$

2. Infer reductions in the encoding corresponding to:

   $$\pi(b().c() \mid a(X).(X \mid b(n).\pi()) \rightarrow c() \mid b(c) \mid b(n).\pi()) \rightarrow c() \mid \pi()) \rightarrow 0$$

3. Consider a Local Area Higher-order $\pi$:

   $$\text{net}[\pi(b().c() \mid a(X).(X \mid b(n).\pi())]$$

   $$\rightarrow \text{net}[\pi(c()) \mid \text{host}[b]()]$$

   $$\rightarrow \text{net}[\pi(c()) \mid \text{host}()] \rightarrow \text{net}[\text{host}[0] \mid \text{host}[0]]$$

   and assume that $a$ and $c$ operate at net level and $b$ operates at host level. Explain what happens if we apply the encoding of 1. to the process in 3.

Exercise — Higher-order Encoding of Replication

Try to infer reactions of the higher-order encoding of replication.

- Encoding of replication

  $$[!P] = (\nu a)(D \mid \pi(P \mid D))$$

  $$D = a(X).(X \mid \pi(X)),$$ where $a \notin \text{fn}(P)$

- Why do we have to send $D$ with $P$?

All Done

If you have done all the exercises, come ask me for more ;)

Mandatory Exercise

1. Apply the encoding of Higher-order $\pi$-calculus into first-order $\pi$-calculus to the following expression:

   $$\pi(c).c() \mid a(X).(X \mid b(n).\pi())$$

2. Infer reductions in the encoding corresponding to:

   $$\pi(b().c() \mid a(X).(X \mid b(n).\pi()) \rightarrow c() \mid b(c) \mid b(n).\pi()) \rightarrow c() \mid \pi()) \rightarrow 0$$

3. Consider a Local Area Higher-order $\pi$:

   $$\text{net}[\pi(b().c() \mid a(X).(X \mid b(n).\pi())]$$

   $$\rightarrow \text{net}[\pi(c()) \mid \text{host}[b]()]$$

   $$\rightarrow \text{net}[\pi(c()) \mid \text{host}()] \rightarrow \text{net}[\text{host}[0] \mid \text{host}[0]]$$

   and assume that $a$ and $c$ operate at net level and $b$ operates at host level. Explain what happens if we apply the encoding of 1. to the process in 3.