

Overview of the Lecture

Higher-order, Mobility and Locations *Model-based Design of Distributed and Mobile Systems Lecture 10: Spring 2005*

Thomas Hildebrandt

hilde -at- itu.dk

Department of Theoretical Computer Science
IT University of Copenhagen

Model-based Design of Distributed and Mobile Systems Lecture 10: Spring 2005 – p.1/21

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

Model-based Design of Distributed and Mobile Systems Lecture 10: Spring 2005 – p.2/21

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*.

Model-based Design of Distributed and Mobile Systems Lecture 10: Spring 2005 – p.3/21

Higher-order π -calculus

The ideas of Bent Thomsen and Davide Sangiorgi

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

- New rule for reaction

$$\text{REACTHO} \frac{}{(a(X).P + P') \mid (\bar{a}(Q).R + R') \longrightarrow P\{Q/X\} \mid R}$$

Examples:

- $\bar{a}(R).P \mid a(X).X \longrightarrow P \mid R$
- $\bar{a}(R).P \mid a(X).0 \longrightarrow P$

Model-based Design of Distributed and Mobile Systems Lecture 10: Spring 2005 – p.4/21

Copying processes

Since an agent variable can occur several times in a agent, we can now **copy agents**

$$a(X).(X \mid X) \mid \bar{a}(Q).R \longrightarrow Q \mid Q \mid R$$

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

- Encoding of replication

$$\begin{aligned} \llbracket !P \rrbracket &= (\nu a)(D \mid \bar{a}(P \mid D)) \\ D &= a(X).(X \mid \bar{a}(X)), \text{ where } a \notin fn(P) \end{aligned}$$

- Why do we have to send D with P ?

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. $\llbracket P \mid Q \rrbracket = \llbracket P \rrbracket \mid \llbracket Q \rrbracket$
 - 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 $\llbracket P \rrbracket \sim \llbracket Q \rrbracket$

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

Higher-order into First-order

- First-order π -calculus is **expressive enough** to encode higher-order π -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

$$\llbracket \bar{a}(P).Q \rrbracket = (\nu p)\bar{a}(p).(\llbracket Q \rrbracket \mid !p.\llbracket P \rrbracket), \text{ where } p \notin fn(P, Q)$$

$$\llbracket a(X).P \rrbracket = a(x).\llbracket P \rrbracket$$

$$\llbracket X \rrbracket = \bar{x}$$

Example

In HO π : $\bar{a}(R).P \mid a(X).(X \mid X) \longrightarrow P \mid R \mid R$

In π -calculus: $(\nu r)\bar{a}(r).(\llbracket P \rrbracket \mid !r.\llbracket R \rrbracket) \mid a(x).(\bar{x} \mid \bar{x}) \longrightarrow$

$(\nu r)(\llbracket P \rrbracket \mid !r.\llbracket R \rrbracket \mid (\bar{r} \mid \bar{r})) \longrightarrow$

$(\nu r)(\llbracket P \rrbracket \mid !r.\llbracket R \rrbracket \mid \llbracket R \rrbracket \mid \bar{r}) \longrightarrow \sim \llbracket P \rrbracket \mid \llbracket R \rrbracket \mid \llbracket R \rrbracket$

Summary on Higher-order π

- 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 π -calculus ($la\pi$) [Chothia, PhD thesis]

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

Client:

$(\nu c)(\overline{server}\langle finger, c \rangle$ ← ask server
 $| c(x).\overline{print}\langle x \rangle$ ← print response

Server:

$server(service, reply).\overline{service}\langle reply \rangle$ ← inet daemon
 $!finger(reply).\overline{reply}\langle users \rangle$ ← finger daemon
 $!time(reply).\overline{reply}\langle now \rangle$ ← 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 π encoded in (polyadic) π

- Again, the standard first-order (polyadic) π -calculus is **expressive enough** to encode higher-order π -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 $\overline{a}\langle b \rangle$ becomes $\overline{e}\langle a, b \rangle$ where e is a name corresponding to the local area on which a operates
- Encoding of input: Fetch input from ether and "match or resend"

$$[[a(b).P]] = recXe(x, b).if\ x = a\ then\ [[P]]\ else\ (\overline{e}\langle x, b \rangle \mid X)$$

Can we encode $la\pi$ into monadic π ?

Can the encodings of Higher-order and Local Area be combined to an encoding of $HOLA\pi$ into π ?

Local Area π -calculus

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

Client:

$Hilde = host[(\nu c)(\overline{server}\langle finger, c \rangle$ ← ask server
 $| c(x).\overline{print}\langle x \rangle]$ ← print response

Server:

$ITU = host[server(service, reply).\overline{service}\langle reply \rangle$ ← inet daemon
 $!finger(reply).\overline{reply}\langle users \rangle$ ← finger daemon
 $!time(reply).\overline{reply}\langle now \rangle]$ ← time daemon

System:

$net[Hilde \mid ITU]$

Names:

$server$ and c operate at *net* level;
 $finger, time$ and $print$ operate at *host* level;

Summary on Local Area π

- 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

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 $\bar{a}(P).Q$ prefixes, and *process* variables X as in CHOCS and HO π
- Higher-order *take* $\bar{a}(X).Q$ and (reactive) *resource* $a(P).Q$ prefixes
- *Nested names* e.g. $a.b$ and new rules for reaction, e.g.

$$\frac{}{\bar{a}.b(X).P \mid (a\langle b(Q).Q' \mid Q'' \rangle) \longrightarrow P\{Q/X\} \mid a\langle Q' \mid Q'' \rangle}$$

$$\frac{}{\bar{a}.b(R).P \mid (a\langle b(X).Q \mid Q' \rangle) \longrightarrow P \mid a\langle Q\{R/X\} \mid Q' \rangle}$$

- Full Homer also adds *active resource* $a[P].Q$ prefixes and the rule

$$\frac{P \longrightarrow P'}{a[P].Q \longrightarrow a[P'].Q}$$

Model-based Design of Distributed and Mobile Systems Lecture 10: Spring 2005 – p.13/21

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

Model-based Design of Distributed and Mobile Systems Lecture 10: Spring 2005 – p.15/21

π and spi into Homer_r

- First-order π -calculus can be encoded into Homer_r, *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_r
- Principle of encoding: Encode encryption by nesting.

Model-based Design of Distributed and Mobile Systems Lecture 10: Spring 2005 – p.14/21

Mobile Ambients

L. Cardelli and A. Gordon, Journal of TCS 2000:

- Subjective, linear mobility between explicit nested locations (ambients)
- $a[in\ b.P \mid P'] \mid b[Q] \longrightarrow b[a[P \mid P'] \mid Q]$
- $a[b[out\ a.P \mid P'] \mid Q] \longrightarrow b[P \mid P'] \mid a[Q]$
- $open\ b.P \mid b[Q] \longrightarrow P \mid Q$
- Ambients are active (as in Homer):

$$\frac{P \longrightarrow P'}{a[P] \longrightarrow a[P']}$$

Model-based Design of Distributed and Mobile Systems Lecture 10: Spring 2005 – p.16/21

Ambient Examples

- Locks:
 $acquire\ n.P =_{def} open\ n.P$
 $release\ n.P =_{def} n[] \mid P$
- Firewall: $(a)a[P]$
- CCS interaction, Renaming, Turing Machines...

Mandatory Exercise

1. Apply the encoding of Higher-order π -calculus into first-order π -calculus to the following expression
 $\bar{a}\langle\bar{b}\langle c\rangle\rangle.c() \mid a(X).(X \mid b(n).\bar{n}\langle\rangle)$
2. Infer reductions in the encoding corresponding to
 $\bar{a}\langle\bar{b}\langle c\rangle\rangle.c() \mid a(X).(X \mid b(n).\bar{n}\langle\rangle) \longrightarrow c() \mid \bar{b}\langle c\rangle \mid b(n).\bar{n}\langle\rangle \longrightarrow c() \mid \bar{c}\langle\rangle \longrightarrow 0$
3. Consider a Local Area Higher-order pi:
 $net[host[\bar{a}\langle\bar{b}\langle c\rangle\rangle.c()] \mid host[a(X).(X \mid b(n).\bar{n}\langle\rangle)]]$
 $\longrightarrow net[host[c()] \mid host[\bar{b}\langle c\rangle \mid b(n).\bar{n}\langle\rangle]]$
 $\longrightarrow net[host[c()] \mid host[\bar{c}\langle\rangle]] \longrightarrow net[host[0] \mid host[0]]$
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

$$\llbracket !P \rrbracket = (\nu a)(D \mid \bar{a}\langle P \mid D \rangle)$$
$$D = a(X).(X \mid \bar{a}\langle X \rangle), \text{ where } a \notin fn(P)$$

- Why do we have to send D with P ?

All Done

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