A Denotational Study of Mobility

Joël-Alexis Bialkiewicz and Frédéric Peschanski

November 2, 2009
Introduction

Two Main Points of View on Modelling Processes

- **Operational POV** ($\pi$-calculus...)
  - Low level, double-edged: easy mobility but difficult to abstract
  - unsettled theory so many variants
  - issues with compositionality: bound prefixes and guards
  - denotations exist but not practical

- **Denotational POV** (CSP)
  - denotational (tr, fail, div) and compositional by design
  - supports refinement
  - but no easy way to account for mobility

Our Approach: Mobility in a Denotational Way

- Heavily inspired by CSP but integrated model (decorated traces)
- $\pi$-like mobility but compositional $\implies$ fully denotational model
- Support for refinement
Introduction

Two Main Points of View on Modelling Processes

- **Operational POV (\(\pi\)-calculus...)**
  - Low level, double-edged: easy mobility but difficult to abstract
  - unsettled theory so many variants
  - issues with compositionality: bound prefixes and guards
  - denotations exist but not practical

- **Denotational POV (CSP)**
  - denotational (tr, fail, div) and compositional by design
  - supports refinement
  - but no easy way to account for mobility

Our Approach: Mobility in a Denotational Way

- Heavily inspired by CSP but integrated model (decorated traces)
- \(\pi\)-like mobility but compositional \(\Rightarrow\) fully denotational model
- Support for refinement
Outline

1. Basics: Locations
2. Mobility
3. Equivalence & Refinement
Outline

1. Basics: Locations
2. Mobility
3. Equivalence & Refinement
The problem

- Full representation of behaviour? branching structure (LTS)
- Set of process traces: information lost
- Traces + failures, divergences: hard to introduce mobility

What we wanted

- Traces but with as much information as the LTS
- How: link observations to where and when in LTS
  \[ \Rightarrow \textit{locations} ! \]
- LTS can be rebuilt from decorated traces: no information lost
Basic Example

The basics

Observation (::location): input channel? output channel!value, or ✓
Location: origin: $\epsilon$, next: $\triangleright$ and choice: $\diamondnumbr$ of branch number
weak variants $\tilde{\triangleright}$ and $\tilde{\diamond}\jmath$

Process (not mobile)

$\text{coin}?.(\text{button1}?.\text{out}!\text{tea} + \text{coin}?.\text{button2}?.\text{out}!\text{coffee})$

What does this process do?

Behaviour

<table>
<thead>
<tr>
<th>LTS</th>
<th>Traces</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{coin}?$</td>
<td>{ \langle \text{coin}?:\triangleright, \text{button1}?::\diamond_2^1, \text{out}!\text{tea}::\triangleright \rangle, \langle \text{coin}?:\triangleright, \text{coin}?:::\diamond_2^2, \text{button2}?::\triangleright, \text{out}!\text{coffee}::\triangleright \rangle }</td>
</tr>
</tbody>
</table>
Basic Example

The basics

Observation (::location): input channel? output channel!value, or ✓
Location: origin: ∈, next: ▶ and choice: ◊_{branch number}, weak variants \tilde{\triangleleft} and \tilde{\triangledown}_i

Process (not mobile)

\textit{coin}?(button1?.out!tea + coin?.button2?.out!coffee)

What does this process do?

Behaviour

```
\begin{align*}
\text{coin?} & \rightarrow \text{coin?} \\
\text{out!tea} & \rightarrow \text{button2?} \\
\text{out!coffee} & \rightarrow
\end{align*}
```

Traces

```
\{ \\
\langle \text{coin}?:\triangledown, \text{button1}?::\diamondsuit_1, \text{out!tea}::\triangledown \rangle, \\
\langle \text{coin}?:\triangledown, \text{coin}?::\diamondsuit_2, \text{button2}?::\triangledown, \text{out!coffee}::\triangledown \rangle \\
\}
```
Basic Example

The basics

Observation (::location): input channel? output channel!value, or ✓
Location: origin: $\epsilon$, next: $\triangleright$ and choice: $\diamond^{\text{number of branches}}_{\text{branch number}}$, weak variants $\sim\triangleright$ and $\sim_{\text{i}}$

Process (not mobile)

$\text{coin}?(\text{button1}? \cdot \text{out!tea} + \text{coin}?.\text{button2}? \cdot \text{out!coffee})$

What does this process do?

Behaviour

![LTS diagram]

Traces

\[
\{ \\
\langle \text{coin}?:\triangleright, \text{button1}?:\diamond^2_1, \text{out!tea}:\triangleright \rangle, \\
\langle \text{coin}?:\triangleright, \text{coin}?:\diamond^2_2, \text{button2}?:\triangleright, \text{out!coffee}:\triangleright \rangle \\
\}
\]
Basic Example

The basics

Observation (::location): input \textit{channel}? output \textit{channel}!value, or \checkmark
Location: origin: $\epsilon$, next: $\triangleright$ and choice: $\lozenge_{\text{branch number}}$, weak variants $\tilde{\triangleright}$ and $\tilde{\lozenge}_i$

Process (not mobile)

$\text{coin}?(button1?.out!tea + coin?.button2?.out!coffee)$

What does this process do?

Behaviour

\begin{itemize}
  \item LTS
  \begin{itemize}
    \item $\text{coin}$
    \item $\text{button1?}$
    \item $\text{out!tea}$
    \item $\text{button2?}$
    \item $\text{out!coffee}$
  \end{itemize}

  \item Traces
  \[
  \begin{aligned}
  \{ & \langle \text{coin}?:\triangleright, \text{button1}?:\lozenge_1^2, \text{out!tea}?:\triangleright \rangle, \\
  & \langle \text{coin}?:\triangleright, \text{coin}?:\lozenge_2^2, \text{button2}?:\triangleright, \text{out!coffee}?:\triangleright \rangle \}
  \end{aligned}
  \]
\end{itemize}
Basic Example

The basics

Observation (::location): input channel?output channel!value, or ✓
Location: origin: ε, next: ▼ and choice: ◊_{branch number}, weak variants ˜▼ and ˜◊_i

Process (not mobile)

coin?.(button1?.out!tea + coin?.button2?.out!coffee)

What does this process do?

Behaviour

Traces

\{
  \langle coin?::▼, button1?::◊_{21}, out!tea::▼ \rangle,
  \langle coin?::▼, coin?::◊_{22}, button2?::▼, out!coffee::▼ \rangle
\}
**Basic Example**

**The basics**

Observation (移民地点): input channel? output channel!value, or \( \square \)
Location: origin: \( \epsilon \), next: \( \triangleright \) and choice: \( \diamond \) number of branches, weak variants \( \sim \) and \( \tilde{\diamond} \)

**Process (not mobile)**

\( \text{coin?.(button1?.out!tea + coin?.button2?.out!coffee)} \)

What does this process do?

**Behaviour**

LTS

\( \text{coin?} \)

\( \text{coin?} \)

\( \text{button2?} \)

\( \text{out!tea} \)

\( \text{out!coffee} \)

**Traces**

\( \{ \langle \text{coin?}::\triangleright, \text{button1?}::\diamond_1^2, \text{out!tea}::\triangleright \rangle, \langle \text{coin?}::\triangleright, \text{coin?}::\diamond_2^2, \text{button2?}::\triangleright, \text{out!coffee}::\triangleright \rangle \} \)
Basic Example

The basics

Observation (::location): input channel? output channel!value, or ✓
Location: origin: ϵ, next: △ and choice: △number of branchesi, weak variants △ and △i

Process (not mobile)

coin?.(button1?.out!tea + coin?.button2?.out!coffee)

What does this process do?

Behaviour

LTS

Traces

\{ (coin?::△, button1?::⋄21, out!tea::△), (coin?::△, coin?::⋄22, button2?::△, out!coffee::△) \}
Basic Example

The basics

Observation (::location): input channel? output channel!value, or ✓
Location: origin: $\epsilon$, next: $\triangleright$ and choice: $\Diamond$ \textit{number of branches}, weak variants $\widetilde{\triangleright}$ and $\widetilde{\Diamond}^j$

Process (not mobile)

$\text{coin}?.(\text{button1?}.\text{out!tea} + \text{coin}?.\text{button2?}.\text{out!coffee})$

Which locations why? What is an absolute location?

Behaviour

\[
\begin{array}{c}
\text{LTS} \\
\begin{array}{c}
\epsilon \\
\text{coin?} \\
\Diamond_1 \text{button1?} \\
\Diamond_2 \text{coin?} \\
\triangleright \text{out!tea} \\
\downarrow \\
\triangleright \text{button2?} \\
\downarrow \\
\text{out!coffee} \\
\end{array}
\end{array}
\]

Traces

\[
\{ \langle \text{coin?}::\triangleright, \text{button1?}::\Diamond_1, \text{out!tea}::\triangleright \rangle, \\
\langle \text{coin?}::\triangleright, \text{coin?}::\Diamond_2, \text{button2?}::\triangleright, \text{out!coffee}::\triangleright \rangle \}
\]
Outline

1. Basics: Locations
2. Mobility
3. Equivalence & Refinement
About Mobility

Physical vs Logical Mobility

A process is mobile if it changes neighbours

How Can a Process Change Neighbours?

\[ \nu c \]

A

B

C

\[ \nu d \]

d

\nu
About Mobility

Physical vs Logical Mobility

A process is mobile if it changes neighbours

How Can a Process Change Neighbours?

\[ \nu c \quad \text{c} \quad \nu d \]

A process moves from location c to location d.
About Mobility

Physical vs Logical Mobility

A process is mobile if it changes neighbours

How Can a Process Change Neighbours?
About Mobility

Physical vs Logical Mobility

A process is mobile if it changes neighbours

How Can a Process Change Neighbours?
About Mobility

Physical vs Logical Mobility

A process is mobile if it changes neighbours

How Can a Process Change Neighbours?
The Modeling Problems

Two main problems
- Binders
- Guards

Binders in mobile languages
- Binders: dynamic names (escaped names and inputs)
- \(\pi\)-calculus operational, mixes free and bound names
- Solution: binders are uniquely identified by when/where created
- Advantage: fresh by construction, avoid \(\alpha\)-conversion issues

Guards
- Reminder: \([\varphi]P\) means if \(\varphi\) then \(P\)
- Not observations, but necessary for compos. Where do they go?
- Solution: in locations
Example 2

Process with guards and extrusion

\((\nu \text{ in}) \text{request}!\text{in.in}\text{?out.}\)
\(((\text{out} = \text{stop}) \text{SKIP} + [\text{out} \neq \text{stop}] \text{Communication(in, out)})\)

What does this process do?

Behaviour

LTS

\([\rho. = \text{stop}]\checkmark\)
\([\rho. \neq \text{stop}]\alpha_1\)

Traces

\{ \langle \text{request}!\nu_{\epsilon\triangleright}, \nu_{\epsilon\triangleright}\text{?}\triangleright, \checkmark::(\rho_{\epsilon\triangleright\triangleright} = \text{stop}, \diamondsuit_1^2) \rangle, \langle \text{request}!\nu_{\epsilon\triangleright}, \nu_{\epsilon\triangleright}\text{?}\triangleright, \alpha_1::(\rho_{\epsilon\triangleright\triangleright} \neq \text{stop}, \diamondsuit_2^2), \ldots \rangle, \ldots \} \}
Example 2

Process with guards and extrusion

\((\nu \text{ in})\text{request!in.in?out.}\)

\([\text{out} = \text{stop}]\text{SKIP} + [\text{out} \neq \text{stop}]\text{Communicate(in, out)})

Extruded names: \(\nu_{\text{where}}\)

**Behaviour**

\[\begin{align*}
\text{LTS} & \\
\nu \cdot = \text{stop} \& & \nu \cdot \neq \text{stop} \\
\text{Traces} & \\
\{ & \langle \text{request!}\nu_{\in}\cdot, \text{out}\cdot, \text{out}\cdot, \text{out}\cdot = \text{stop}, \alpha_{1} = (\rho_{\in}, \rho_{\in}, \rho_{\in}) \rangle, \\
& \langle \text{request!}\nu_{\in}\cdot, \text{out}\cdot, \text{out}\cdot, \text{out}\cdot \neq \text{stop}, \alpha_{1} = (\rho_{\in}, \rho_{\in}, \rho_{\in}) \rangle, \\
& \alpha_{1} = \ldots \}
\end{align*}\]
Example 2

Process with guards and extrusion

\((\nu \text{ in}) \text{request!in.in?out.} \)  
\(([\text{out} = \text{stop}] \text{SKIP} + [\text{out} \neq \text{stop}] \text{Communicate(in, out)})\)

Input observations have no object; received names: \(\rho_{\text{where}}\)

Behaviour

\[
\begin{array}{c}
\text{LTS} \\
\begin{tikzpicture}
  \node (req) at (0,0) [circle,fill,inner sep=2pt] {\text{request!}\nu};
  \node (out) at (-1,-1) [circle,fill,inner sep=2pt] {\nu};
  \node (stop) at (-2,-2) [circle,fill,inner sep=2pt] {\rho = \text{stop}};
  \node (notStop) at (-2,-3) [circle,fill,inner sep=2pt] {\rho \neq \text{stop}};
  \draw[->] (req) -- (out);
  \draw[->] (out) -- (stop);
  \draw[->] (out) -- (notStop);
\end{tikzpicture}
\end{array}
\]

Traces

\[
\{ \\
\langle \text{request!}\nu_{\epsilon\Delta}::\Delta, \nu_{\epsilon\Delta}::\Delta, \nu_{\epsilon\Delta}::\Delta, \gamma::(\rho_{\epsilon\Delta\Delta} = \text{stop}, \diamond_{1}^{2}), \rangle, \\
\langle \text{request!}\nu_{\epsilon\Delta}::\Delta, \nu_{\epsilon\Delta}::\Delta, \alpha_{1}::(\rho_{\epsilon\Delta\Delta} \neq \text{stop}, \diamond_{2}^{2}), \ldots \rangle, \\
\ldots \}
\]
Example 2

Process with guards and extrusion

\[(\nu \text{ in}) \text{request!in.in?out.} + \nu = \text{stop}] \text{SKIP} + \nu \neq \text{stop}] \text{Communicate(in, out)}\]

In traces the guard of an observation prefixes its location

![LTS diagram]

\[\{ \langle \text{request!\nu} \in \triangleright, \nu \in \triangleright?\in \triangleright, \checkmark \in \triangleright: (\rho \in \triangleright \triangleright = \text{stop}, \diamond_1^2), \rangle, \langle \text{request!\nu} \in \triangleright, \nu \in \triangleright?\in \triangleright, \alpha_1 \in \triangleright: (\rho \in \triangleright \triangleright \neq \text{stop}, \diamond_2^2), \ldots, \rangle, \ldots \}\]
Example 2

Process with guards and extrusion

\((\nu \text{ in}) \text{request}!\text{in.in?out.}\)
\(([\text{out} = \text{stop}]\text{SKIP} + [\text{out} \neq \text{stop}]\text{Communicate(in, out)})\)

In traces the guard of an observation prefixes its location

\[\begin{array}{c}
\text{LTS} \\
\nu_\bullet !. \\
\nu_\bullet ?. \\
[\rho_\bullet = \text{stop}]\checkmark \\
[\rho_\bullet \neq \text{stop}]\alpha_1
\end{array}\]

\[\begin{array}{c}
\text{Traces} \\
\{ \langle \text{request}!\nu_\epsilon \triangleright, \nu_\epsilon \triangleright, \checkmark (\rho_\epsilon \triangleright = \text{stop}, \diamond_1^2) \rangle, \\
\langle \text{request}!\nu_\epsilon \triangleright, \nu_\epsilon \triangleright, \alpha_1 (\rho_\epsilon \triangleright \neq \text{stop}, \diamond_2^2), \ldots \rangle, \\
\ldots \}
\end{array}\]
Outline

1. Basics: Locations
2. Mobility
3. Equivalence & Refinement
Dealing with redundancy

- Problem: model very fine-grained
- Solution: rewrite rules to trim redundancy

Theorem

Let $T$ be a trace set. Suppose $T_1$ and $T_2$ such that $T \nrightarrow^*$ $T_1 \nrightarrow^*$ and $T \nrightarrow^*$ $T_2 \nrightarrow^*$. Then $T_1 = T_2 = \hat{T}$.

Interest

- Equivalence checking: normalise then test isomorphism
- Much simpler than existing equivalence checking for mobility
- Only possible because no binders in semantic
Example

\[ P = \alpha.(\beta + \beta) \]
\[ \text{traces}(P) = \{ \langle \alpha::\triangledown, \beta::\diamondsuit \rangle_1, \langle \alpha::\triangledown, \beta::\diamondsuit \rangle_2 \} \]

\[ Q = \alpha.\beta + [a = x \land a \neq x] \gamma \]
\[ \text{traces}(Q) = \{ \langle \alpha::\diamondsuit \rangle_1, \langle \gamma::(a = x \land a \neq x, \diamondsuit) \rangle_2 \} \]
Example

\[ P = \alpha.(\beta + \beta) \]

\[ \text{traces}(P) = \{ \langle \alpha::\triangledown, \beta::\Diamond^2_1 \rangle, \langle \alpha::\triangledown, \beta::\Diamond^2_2 \rangle \} \]

\[ Q = \alpha.\beta + [a = x \land a \neq x]\gamma \]

\[ \text{traces}(Q) = \{ \langle \alpha::\Diamond^2_1, \beta::\triangledown \rangle, \langle \gamma::(a = x \land a \neq x, \Diamond^2_2) \rangle \} \]
Example

\[ P = \alpha.(\beta + \beta) \]

\[ \text{traces}(P) = \{ \langle \alpha::\triangledown, \beta::\diamondsuit^2_1 \rangle, \langle \alpha::\triangledown, \beta::\diamondsuit^2_2 \rangle \} \]

\[ \text{traces}(P) \xrightarrow{\text{merge}} \{ \langle \alpha::\triangledown, \beta::\triangledown \rangle \} \]

\[ Q = \alpha.\beta + [a = x \land a \neq x]\gamma \]

\[ \text{traces}(Q) = \{ \langle \alpha::\triangledown^2_1, \beta::\triangledown \rangle, \langle \gamma::(a = x \land a \neq x, \diamondsuit^2_2) \rangle \} \]
Example

\[ P = \alpha.(\beta + \beta) \]
\[ \text{traces}(P) = \{ \langle \alpha::\triangledown, \beta::\circadian_1 \rangle, \langle \alpha::\triangledown, \beta::\circadian_2 \rangle \} \]
\[ \text{traces}(P) \xrightarrow{\text{merge}} \{ \langle \alpha::\triangledown, \beta::\triangledown \rangle \} \]

\[ Q = \alpha.\beta + [a = x \land a \neq x] \gamma \]
\[ \text{traces}(Q) = \{ \langle \alpha::\circadian_1^2, \beta::\triangledown \rangle, \langle \gamma::(a = x \land a \neq x, \circadian_2^2) \rangle \} \]
\[ \text{traces}(Q) \xrightarrow{\text{false}} \{ \langle \alpha::\triangledown, \beta::\triangledown \rangle \} \]
Example

\[ P = \alpha.(\beta + \beta) \]

\[ \text{traces}(P) = \{ \langle \alpha::\triangleright, \beta::\bowtie_1^2 \rangle, \langle \alpha::\triangleright, \beta::\bowtie_1^2 \rangle \} \]

\[ \text{traces}(P) \xrightarrow{\text{merge}} \{ \langle \alpha::\triangleright, \beta::\triangleright \rangle \} \]

\[ Q = \alpha.\beta + [a = x \land a \neq x] \gamma \]

\[ \text{traces}(Q) = \{ \langle \alpha::\bowtie_1^2, \beta::\triangleright \rangle, \langle \gamma::(a = x \land a \neq x, \bowtie_2^2) \rangle \} \]

\[ \text{traces}(Q) \xrightarrow{\text{false}} \{ \langle \alpha::\triangleright, \beta::\triangleright \rangle \} \]
What is the delayed sum?

- The way to refinement
- Strict generalisation of process sum
- Grafting any behaviour anywhere in branching structure
- Two parameters: a location and a substitution from symbols to special names

Delayed Sum Example

\[ P \overset{\text{def}}{=} \alpha.\beta + \gamma \]
\[ Q \overset{\text{def}}{=} \delta.\alpha \]
\[ P +_{\epsilon_{\diamond}}^{\text{Id}} Q = \alpha.(\beta + \delta.\alpha) + \gamma \]
Delayed Sum

What is the delayed sum?
- The way to refinement
- Strict generalisation of process sum
- Grafting any behaviour anywhere in branching structure
- Two parameters: a location and a substitution from symbols to special names

Delayed Sum Example

\[ P \overset{\text{def}}{=} \alpha \cdot \beta + \gamma \quad Q \overset{\text{def}}{=} \delta \cdot \alpha \]

\[ P +_{\epsilon \diamond 1}^{Id} Q = \alpha \cdot (\beta + \delta \cdot \alpha) + \gamma \]
Delayed Sum

What is the delayed sum?
- The way to refinement
- Strict generalisation of process sum
- Grafting any behaviour anywhere in branching structure
- Two parameters: a location and a substitution from symbols to special names

Delayed Sum Example

\[ P \overset{\text{def}}{=} \alpha \cdot \beta + \gamma \quad Q \overset{\text{def}}{=} \delta.\alpha \quad P +^{\text{Id}_{\epsilon_1}}_{\epsilon_2} Q = \alpha.(\beta + \delta.\alpha) + \gamma \]
Delayed Sum

What is the delayed sum?

- The way to refinement
- Strict generalisation of process sum
- Grafting any behaviour anywhere in branching structure
- Two parameters: a location and a substitution from symbols to special names

Delayed Sum Example

\[ P \overset{\text{def}}{=} \alpha \cdot \beta + \gamma \quad Q \overset{\text{def}}{=} \delta \cdot \alpha \quad P +_{\epsilon_1^2}^\text{Id} Q = \alpha \cdot (\beta + \delta \cdot \alpha) + \gamma \]
Refinement

**Definition**

\[ P \sqsubseteq Q \iff \exists \mathcal{RL} = \bigcup_{i=1}^{n} \{(R_i, l_i, \sigma_i)\} \quad \text{s. t.} \]

\[ P = \lozenge Q +_{l_1}^{\sigma_1} R_1 \ldots +_{l_n}^{\sigma_n} R_n \]

**Why?**

- Refinement relation nearly for free
- The delayed sum cannot be compositional... is refinement?
Conclusion

What we did
- CSP vs $\pi$-calculus: a step towards bridging the gap
- Denotational theory for mobility with intuitive refinement
- Operational semantics w/o $\pi$-calculus pitfalls
- Axiomatic semantics
- A Hoare-like logic [LAM09]

What next?
- Finish writing the thesis…
- Proving that refinement is compositional
- Equivalence/refinement checking algorithm