Mobile Escape Analysis for occam-pi

CPA-2009

Fred Barnes

School of Computing, University of Kent, Canterbury

F.R.M.Barnes@kent.ac.uk

http://www.cs.kent.ac.uk/~frmb/

Programming Languages and Systems

University of Kent Computing

Summary

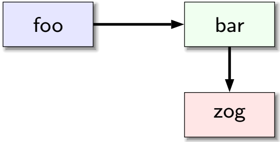
- We've been developing **occam-π** programs for some time now:
  - traditional process-oriented design of concurrent **processes** and **communication**
  - **dynamics** added from Milner's π-calculus: **mobile** data, channels and processes
  - real applications for complex systems simulation (**CoSMoS**) and operating systems (**RMoX**)
- Semantics from **CSP** [1, Hoare-1985], on which the original occam language was based:
  - provides **formal reasoning** for parallel processes and their interactions
- We also have CSP models for mobile data, channels and processes:
  - largely for an understanding of their **operational** behaviour
- What we do not yet have:
  - a **denotational** and **compositional** understanding of how mobile systems **evolve**

Summary

- We've been developing **occam-π** programs for some time now:
  - traditional process-oriented design of concurrent **processes** and **communication**
  - **dynamics** added from Milner's π-calculus: **mobile** data, channels and processes
  - real applications for complex systems simulation (**CoSMoS**) and operating systems (**RMoX**)
- Semantics from **CSP** [1, Hoare-1985], on which the original occam language was based:
  - provides **formal reasoning** for parallel processes and their interactions
- We also have CSP models for mobile data, channels and processes:
  - largely for an understanding of their **operational** behaviour
- What we do not yet have:
  - a **denotational** and **compositional** understanding of how mobile systems **evolve**

Summary

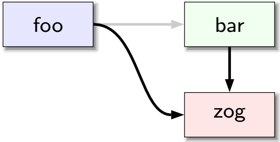
- We've been developing **occam-π** programs for some time now:
  - traditional process-oriented design of concurrent **processes** and **communication**
  - **dynamics** added from Milner's π-calculus: **mobile** data, channels and processes
  - real applications for complex systems simulation (**CoSMoS**) and operating systems (**RMoX**)



- Semantics from **CSP** [1, Hoare-1985], on which the original occam language was based:
  - provides **formal reasoning** for parallel processes and their interactions
- We also have CSP models for mobile data, channels and processes:
  - largely for an understanding of their **operational** behaviour
- What we do not yet have:
  - a **denotational** and **compositional** understanding of how mobile systems **evolve**

Summary

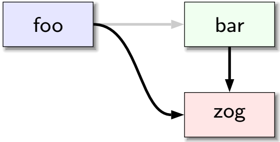
- We've been developing **occam-π** programs for some time now:
  - traditional process-oriented design of concurrent **processes** and **communication**
  - **dynamics** added from Milner's π-calculus: **mobile** data, channels and processes
  - real applications for complex systems simulation (**CoSMoS**) and operating systems (**RMoX**)



- Semantics from **CSP** [1, Hoare-1985], on which the original occam language was based:
  - provides **formal reasoning** for parallel processes and their interactions
- We also have CSP models for mobile data, channels and processes:
  - largely for an understanding of their **operational** behaviour
- What we do not yet have:
  - a **denotational** and **compositional** understanding of how mobile systems **evolve**

Summary

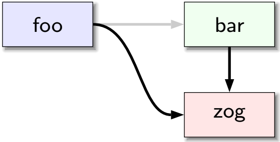
- We've been developing **occam-π** programs for some time now:
  - traditional process-oriented design of concurrent **processes** and **communication**
  - **dynamics** added from Milner's π-calculus: **mobile** data, channels and processes
  - real applications for complex systems simulation (**CoSMoS**) and operating systems (**RMoX**)



- Semantics from **CSP** [1, Hoare-1985], on which the original occam language was based:
  - provides **formal reasoning** for parallel processes and their interactions
- We also have CSP models for mobile data, channels and processes:
  - largely for an understanding of their **operational** behaviour
- What we do not yet have:
  - a **denotational** and **compositional** understanding of how mobile systems **evolve**

Summary

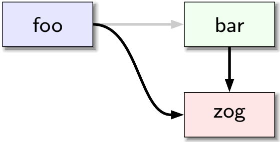
- We've been developing **occam-π** programs for some time now:
  - traditional process-oriented design of concurrent **processes** and **communication**



  - **dynamics** added from Milner's π-calculus: **mobile** data, channels and processes
  - real applications for complex systems simulation (**CoSMoS**) and operating systems (**RMoX**)
- Semantics from **CSP** [1, Hoare-1985], on which the original occam language was based:
  - provides **formal reasoning** for parallel processes and their interactions
- We also have CSP models for mobile data, channels and processes:
  - largely for an understanding of their **operational** behaviour
- What we do not yet have:
  - a **denotational** and **compositional** understanding of how mobile systems **evolve**

Summary

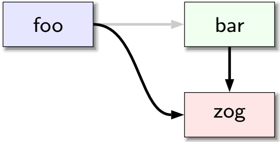
- We've been developing **occam-π** programs for some time now:
  - traditional process-oriented design of concurrent **processes** and **communication**
  - **dynamics** added from Milner's π-calculus: **mobile** data, channels and processes
  - real applications for complex systems simulation (**CoSMoS**) and operating systems (**RMoX**)



- Semantics from **CSP** [1, Hoare-1985], on which the original occam language was based:
  - provides **formal reasoning** for parallel processes and their interactions
- We also have CSP models for mobile data, channels and processes:
  - largely for an understanding of their **operational** behaviour
- What we do not yet have:
  - a **denotational** and **compositional** understanding of how mobile systems **evolve**

Summary

- We've been developing **occam-π** programs for some time now:



  - traditional process-oriented design of concurrent **processes** and **communication**
  - **dynamics** added from Milner's π-calculus: **mobile** data, channels and processes
  - real applications for complex systems simulation (**CoSMoS**) and operating systems (**RMoX**)
- Semantics from **CSP** [1, Hoare-1985], on which the original occam language was based:
  - provides **formal reasoning** for parallel processes and their interactions
- We also have CSP models for mobile data, channels and processes:
  - largely for an understanding of their **operational** behaviour
- What we do not yet have:
  - a **denotational** and **compositional** understanding of how mobile systems **evolve**

Mobile Escape Analysis

- Existing semantic models: **traces**, **failures** and **divergences**.
- New semantic model: **mobility**.
  - primarily interested in how mobiles **move** around a system.
  - to determine the boundaries of any particular mobile item within the **communication graph**.
  - where that graph may be dynamic and **evolve** at run-time.

Traces, Failures, Divergences

- Using a simple occam-π process as an example:

```
PROC id (CHAN INT in?, out!)
  WHILE TRUE
    INT x:
    SEQ
      in ? x
      out ! x
:
```

- Can generate (**automatically** [2, Barnes,Ritson-2009]) a CSP **model** of this process:

$$\mathrm{ID}(\mathit{in}, \mathit{out}) = \mathit{in} \rightarrow \mathit{out} \rightarrow \mathrm{ID}(\mathit{in}, \mathit{out})$$

- And from that the **semantic models**:

$$\mathit{traces}\,\mathrm{ID} = \{\langle\rangle, \langle \mathit{in}\rangle, \langle \mathit{in}, \mathit{out}\rangle, \langle \mathit{in}, \mathit{out}, \mathit{in}\rangle, \ldots\}$$

$$\mathit{failures}\,\mathrm{ID} = \{(\langle\rangle, \{\mathit{out}\}), (\langle \mathit{in}\rangle, \{\mathit{in}\}), (\langle \mathit{in}, \mathit{out}\rangle, \{\mathit{out}\}), \ldots\}$$

$$\mathit{divergences}\,\mathrm{ID} = \{\}$$

Traces, Failures, Divergences

- Using a simple occam-π process as an example:

```
PROC id (CHAN INT in?, out!)
  WHILE TRUE
    INT x:
    SEQ
      in ? x
      out ! x
:
```

- Can generate (**automatically** [2, Barnes,Ritson-2009]) a CSP **model** of this process:

$$\mathrm{ID}(in, out) = in \rightarrow out \rightarrow \mathrm{ID}(in, out)$$

- And from that the **semantic models**:

$$traces\ \mathrm{ID} = \{\langle\rangle, \langle in\rangle, \langle in, out\rangle, \langle in, out, in\rangle, \ldots\}$$

$$failures\ \mathrm{ID} = \{(\langle\rangle, \{out\}), (\langle in\rangle, \{in\}), (\langle in, out\rangle, \{out\}), \ldots\}$$

$$divergences\ \mathrm{ID} = \{\}$$

Traces, Failures, Divergences

- Using a simple occam-π process as an example:

```
PROC id (CHAN INT in?, out!)
  WHILE TRUE
    INT x:
    SEQ
      in ? x
      out ! x
:
```

- Can generate (**automatically** [2, Barnes,Ritson-2009]) a CSP **model** of this process:

$$\mathrm{ID}(\mathit{in}, \mathit{out}) = \mathit{in} \rightarrow \mathit{out} \rightarrow \mathrm{ID}(\mathit{in}, \mathit{out})$$

- And from that the **semantic models**:

$$\begin{aligned}
\mathit{traces}\,\mathrm{ID} &= \{\langle\rangle, \langle \mathit{in}\rangle, \langle \mathit{in}, \mathit{out}\rangle, \langle \mathit{in}, \mathit{out}, \mathit{in}\rangle, \ldots\} \\
\mathit{failures}\,\mathrm{ID} &= \{(\langle\rangle, \{\mathit{out}\}), (\langle \mathit{in}\rangle, \{\mathit{in}\}), (\langle \mathit{in}, \mathit{out}\rangle, \{\mathit{out}\}), \ldots\} \\
\mathit{divergences}\,\mathrm{ID} &= \{\}
\end{aligned}$$

Mobility Analysis

- Similar in concept to the **traces** model – and borrows its syntax.
  - describes what the **mobile behaviour** of a process is.
- For the earlier 'ID' process (which does not involve mobiles):

$$\textit{mobility}\ \mathrm{ID} = \{\}$$

- For an 'MID' process that transports/buffers mobiles:
- Same **traces**, **failures** and **divergences** as before, however:

$$\textit{mobility}\ \mathrm{MID} = \{\textit{in}?^{a}, \textit{out}!^{a}\}$$

Mobility Analysis

- Similar in concept to the **traces** model – and borrows its syntax.
  - describes what the **mobile behaviour** of a process is.
- For the earlier 'ID' process (which does not involve mobiles):

$$\textit{mobility}\ \mathrm{ID} = \{\}$$

- For an 'MID' process that transports/buffers mobiles:
- Same **traces**, **failures** and **divergences** as before, however:

$$\textit{mobility}\ \mathrm{MID} = \{\textit{in}?^{a}, \textit{out}!^{a}\}$$

Mobility Analysis

- Similar in concept to the **traces** model – and borrows its syntax.
  - describes what the **mobile behaviour** of a process is.
- For the earlier 'ID' process (which does not involve mobiles):

$$\mathit{mobility}\ \mathrm{ID} = \{\}$$

- For an 'MID' process that transports/buffers mobiles:

```
PROC mid (CHAN MOBILE THING in?, out!)
  WHILE TRUE
    MOBILE THING x:
    SEQ
      in ? x
      out ! x
:
```

- Same **traces**, **failures** and **divergences** as before, however:

$$\mathit{mobility}\ \mathrm{MID} = \{\mathit{in}?^{a}, \mathit{out}!^{a}\}$$

Mobility Analysis

- Similar in concept to the **traces** model – and borrows its syntax.
  - describes what the **mobile behaviour** of a process is.
- For the earlier 'ID' process (which does not involve mobiles):

$$\textit{mobility}\ \mathrm{ID} = \{\}$$

- For an 'MID' process that transports/buffers mobiles:

```
PROC mid (CHAN MOBILE THING in?, out!)
  WHILE TRUE
    MOBILE THING x:
    SEQ
      in ? x
      out ! x
:
```

- Same **traces**, **failures** and **divergences** as before, however:

$$\textit{mobility}\ \mathrm{MID} = \{\textit{in}?^{a}, \textit{out}!^{a}\}$$

Mobility Sequences

- Like traces, specify what a process **might do**, not necessarily what it **does do** (though the 'ID' processes can only progress one way).
- In general, S is a **set** of **mobility sequences**:

$$S = \{R_1, R_2, ...\}$$

- Where each R is a sequence of **mobile actions**:

$$R = \langle X_1, X_2, X_3, ...\rangle$$

- And each X is either a **mobile input** or a **mobile output**:

$$X_1 = C!^x \qquad , \qquad X_2 = D?^v$$

- Names within a sequence (C, x, D, v) are **bound** within any particular **set** (including formal parameters) – renaming may be required to avoid capture. Other useful operations:

$$\text{concatenation} : \langle X_1, X_2, ...\rangle \,\hat{}\, \langle Y_1, Y_2, ...\rangle = \langle X_1, X_2, ..., Y_1, Y_2, ...\rangle$$

$$\text{restriction} : \langle X_1, C, ...\rangle - \{C\} = \langle X_1, ...\rangle$$

Mobility Sequences

- Like traces, specify what a process **might do**, not necessarily what it **does do** (though the 'ID' processes can only progress one way).
- In general, S is a **set** of **mobility sequences**:

$$S = \{R_1, R_2, ...\}$$

- Where each R is a sequence of **mobile actions**:

$$R = \langle X_1, X_2, X_3, ...\rangle$$

- And each X is either a **mobile input** or a **mobile output**:

$$X_1 = C!^x \qquad , \qquad X_2 = D?^v$$

- Names within a sequence (C, x, D, v) are **bound** within any particular **set** (including formal parameters) – renaming may be required to avoid capture. Other useful operations:

$$\text{concatenation} : \langle X_1, X_2, ...\rangle \hat{\ } \langle Y_1, Y_2, ...\rangle = \langle X_1, X_2, ..., Y_1, Y_2, ...\rangle$$

$$\text{restriction} : \langle X_1, C, ...\rangle - \{C\} = \langle X_1, ...\rangle$$

Mobility Sequences

- Like traces, specify what a process **might do**, not necessarily what it **does do** (though the ‘ID’ processes can only progress one way).
- In general, S is a **set** of **mobility sequences**:

$$S = \{R_1, R_2, ...\}$$

- Where each R is a sequence of **mobile actions**:

$$R = \langle X_1, X_2, X_3, ...\rangle$$

- And each X is either a **mobile input** or a **mobile output**:

$$X_1 = C!^x \qquad , \qquad X_2 = D?^v$$

- Names within a sequence (C, x, D, v) are **bound** within any particular **set** (including formal parameters) – renaming may be required to avoid capture. Other useful operations:

$$\text{concatenation} : \langle X_1, X_2, ...\rangle \,\hat{}\, \langle Y_1, Y_2, ...\rangle = \langle X_1, X_2, ..., Y_1, Y_2, ...\rangle$$

$$\text{restriction} : \langle X_1, C, ...\rangle - \{C\} = \langle X_1, ...\rangle$$

Mobility Sequences

- Like traces, specify what a process **might do**, not necessarily what it **does do** (though the 'ID' processes can only progress one way).
- In general, S is a **set** of **mobility sequences**:

$$S = \{R_1, R_2, ...\}$$

- Where each R is a sequence of **mobile actions**:

$$R = \langle X_1, X_2, X_3, ...\rangle$$

- And each X is either a **mobile input** or a **mobile output**:

$$X_1 = C!^x \qquad , \qquad X_2 = D?^v$$

- Names within a sequence (C, x, D, v) are **bound** within any particular **set** (including formal parameters) – renaming may be required to avoid capture. Other useful operations:

$$\text{concatenation} : \langle X_1, X_2, ...\rangle \mathbin{\hat{}} \langle Y_1, Y_2, ...\rangle = \langle X_1, X_2, ..., Y_1, Y_2, ...\rangle$$

$$\text{restriction} : \langle X_1, C, ...\rangle - \{C\} = \langle X_1, ...\rangle$$

Mobility Sequences

- Like traces, specify what a process **might do**, not necessarily what it **does do** (though the 'ID' processes can only progress one way).
- In general, S is a **set** of **mobility sequences**:

$$S = \{R_1, R_2, ...\}$$

- Where each R is a sequence of **mobile actions**:

$$R = \langle X_1, X_2, X_3, ... \rangle$$

- And each X is either a **mobile input** or a **mobile output**:

$$X_1 = C!^x \qquad , \qquad X_2 = D?^v$$

- Names within a sequence (C, x, D, v) are **bound** within any particular **set** (including formal parameters) – renaming may be required to avoid capture. Other useful operations:

$$\begin{aligned} \text{concatenation} &: \langle X_1, X_2, ... \rangle \,\hat{}\, \langle Y_1, Y_2, ... \rangle = \langle X_1, X_2, ..., Y_1, Y_2, ... \rangle \\ \text{restriction} &: \langle X_1, C, ... \rangle - \{C\} = \langle X_1, ... \rangle \end{aligned}$$

Generating Models of occam-π Programs

- Input, output and assignment are largely straightforward:

- As are choice (ALT, IF, CASE) and parallelism (PAR).
  - simply the **set union** of the different branches.
  - **hiding** is more complex – e.g. as above with 'Lc'.
  - essentially matching **outputs** with **inputs**, and combining those sequences (potentially expansive!)

Generating Models of occam-π Programs

- Input, output and assignment are largely straightforward:

```
PROC P (CHAN MOBILE THING out!)
  MOBILE THING x:
  SEQ
    ... initialise 'x'
    out ! x
:
```

- As are choice (ALT, IF, CASE) and parallelism (PAR).
  - simply the **set union** of the different branches.
  - **hiding** is more complex – e.g. as above with 'Lc'.
  - essentially matching **outputs** with **inputs**, and combining those sequences (potentially expansive!)

Generating Models of occam-π Programs

- Input, output and assignment are largely straightforward:

```
PROC P (CHAN MOBILE THING out!)
  MOBILE THING x:
  SEQ
    ... initialise ‘x’
    out ! x
:
```

$$\mathit{mobility}\ \mathrm{P} = \{\langle \mathit{out}!^{x} \rangle\}$$

- As are choice (ALT, IF, CASE) and parallelism (PAR).
  - simply the **set union** of the different branches.
  - **hiding** is more complex – e.g. as above with ‘Lc’.
  - essentially matching **outputs** with **inputs**, and combining those sequences (potentially expansive!)

Generating Models of occam-π Programs

- Input, output and assignment are largely straightforward:

```
PROC Q (CHAN MOBILE THING in?)
  MOBILE THING y:
  SEQ
    in ? y
    ... use 'y'
:
```

$$\mathit{mobility}\,\mathrm{P} = \{\langle \mathit{out}!^{x}\rangle\}$$

- As are choice (ALT, IF, CASE) and parallelism (PAR).
  - simply the **set union** of the different branches.
  - **hiding** is more complex – e.g. as above with 'Lc'.
  - essentially matching **outputs** with **inputs**, and combining those sequences (potentially expansive!)

Generating Models of occam-π Programs

- Input, output and assignment are largely straightforward:

```
PROC Q (CHAN MOBILE THING in?)
  MOBILE THING y:
  SEQ
    in ? y
    ... use 'y'
:
```

$$\mathit{mobility}\,\mathrm{P} = \{\langle \mathit{out}!^{x} \rangle\}$$

$$\mathit{mobility}\,\mathrm{Q} = \{\langle \mathit{in}?^{y} \rangle\}$$

- As are choice (ALT, IF, CASE) and parallelism (PAR).
  - simply the **set union** of the different branches.
  - **hiding** is more complex – e.g. as above with 'Lc'.
  - essentially matching **outputs** with **inputs**, and combining those sequences (potentially expansive!)

Generating Models of occam-π Programs

- Input, output and assignment are largely straightforward:

```
PROC R (CHAN MOBILE THING in?, out!)
  MOBILE THING v, w:
  SEQ
    in ? v
    w := v
    out ! w
:
```

$$\textit{mobility}\,\mathrm{P} = \{\langle \textit{out}!^{x} \rangle\}$$

$$\textit{mobility}\,\mathrm{Q} = \{\langle \textit{in}?^{y} \rangle\}$$

- As are choice (ALT, IF, CASE) and parallelism (PAR).
  - simply the **set union** of the different branches.
  - **hiding** is more complex – e.g. as above with 'Lc'.
  - essentially matching **outputs** with **inputs**, and combining those sequences (potentially expansive!)

Generating Models of occam-π Programs

- Input, output and assignment are largely straightforward:

```
PROC R (CHAN MOBILE THING in?, out!)
  MOBILE THING v, w:
  SEQ
    in ? v
    w := v
    out ! w
:
```

$$mobility\ \mathrm{P} = \{\langle out!^{x} \rangle\}$$

$$mobility\ \mathrm{Q} = \{\langle in?^{y} \rangle\}$$

$$mobility\ \mathrm{R} = \{\langle in?^{v}, Lc!^{v} \rangle, \langle Lc?^{w}, out!^{w} \rangle\} \setminus \{Lc\}$$

- As are choice (ALT, IF, CASE) and parallelism (PAR).
  - simply the **set union** of the different branches.
  - **hiding** is more complex – e.g. as above with 'Lc'.
  - essentially matching **outputs** with **inputs**, and combining those sequences (potentially expansive!)

Generating Models of occam-π Programs

- Input, output and assignment are largely straightforward:

```
PROC R (CHAN MOBILE THING in?, out!)
  MOBILE THING v, w:
  SEQ
    in ? v
    w := v
    out ! w
:
```

based on the equivalence: `x := y` $\equiv$

```
CHAN INT c:
PAR
  c ! y
  c ? x
```

$$\textit{mobility}\,\mathrm{P} = \{\langle \textit{out}!^{x} \rangle\}$$

$$\textit{mobility}\,\mathrm{Q} = \{\langle \textit{in}?^{y} \rangle\}$$

$$\begin{aligned}\textit{mobility}\,\mathrm{R} &= \{\langle \textit{in}?^{v}, \textit{Lc}!^{v} \rangle, \langle \textit{Lc}?^{w}, \textit{out}!^{w} \rangle\} \setminus \{\textit{Lc}\} \\ &= \{\langle \textit{in}?^{u}, \textit{out}!^{u} \rangle\}\end{aligned}$$

- As are choice (ALT, IF, CASE) and parallelism (PAR).
  - simply the **set union** of the different branches.
  - **hiding** is more complex – e.g. as above with 'Lc'.
  - essentially matching **outputs** with **inputs**, and combining those sequences (potentially expansive!)

Generating Models of occam-π Programs

- Input, output and assignment are largely straightforward:

```
PROC R (CHAN MOBILE THING in?, out!)
  MOBILE THING v, w:
  SEQ
    in ? v
    w := v
    out ! w
:
```

$$\mathit{mobility}\,\mathrm{P} = \{\langle \mathit{out}!^{x}\rangle\}$$

$$\mathit{mobility}\,\mathrm{Q} = \{\langle \mathit{in}?^{y}\rangle\}$$

$$\begin{aligned}\mathit{mobility}\,\mathrm{R} &= \{\langle \mathit{in}?^{v}, \mathit{Lc}!^{v}\rangle, \langle \mathit{Lc}?^{w}, \mathit{out}!^{w}\rangle\} \setminus \{\mathit{Lc}\} \\ &= \{\langle \mathit{in}?^{u}, \mathit{out}!^{u}\rangle\}\end{aligned}$$

based on the equivalence:

```
x := y
```

$\equiv$

```
CHAN INT c:
PAR
  c ! y
  c ? x
```

- As are choice (ALT, IF, CASE) and parallelism (PAR).
  - simply the **set union** of the different branches.
  - **hiding** is more complex – e.g. as above with ‘Lc’.
  - essentially matching **outputs** with **inputs**, and combining those sequences (potentially expansive!)

Generating Models of occam-π Programs

- Input, output and assignment are largely straightforward:

```
PROC R (CHAN MOBILE THING in?, out!)
  MOBILE THING v, w:
  SEQ
    in ? v
    w := v
    out ! w
:
```

$$\mathit{mobility}\ \mathrm{P} = \{\langle \mathit{out}!^{x} \rangle\}$$

$$\mathit{mobility}\ \mathrm{Q} = \{\langle \mathit{in}?^{y} \rangle\}$$

$$\mathit{mobility}\ \mathrm{R} = \{\langle \mathit{in}?^{v}, \mathit{Lc}!^{v} \rangle, \langle \mathit{Lc}?^{w}, \mathit{out}!^{w} \rangle\} \setminus \{\mathit{Lc}\} = \{\langle \mathit{in}?^{u}, \mathit{out}!^{u} \rangle\}$$

based on the equivalence: `x := y` $\equiv$

```
CHAN INT c:
PAR
  c ! y
  c ? x
```

- As are choice (ALT, IF, CASE) and parallelism (PAR).
  - simply the **set union** of the different branches.
  - **hiding** is more complex – e.g. as above with 'Lc'.
  - essentially matching **outputs** with **inputs**, and combining those sequences (potentially expansive!)

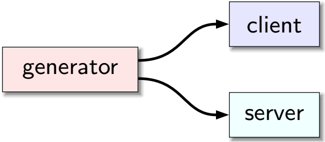
Higher Order Communication

- Things get more interesting when we start moving **channels** around.

- The syntax '$\bar{x}$' represents the **server-end** of a mobile channel-type.
  - **Shared** mobiles are represented as '$x+$'
- The resulting expression here indicates a system in which a mobile 'b' is communicated **internally** over some mobile channel-bundle '$x, \bar{x}$', but which **never escapes**.

Higher Order Communication

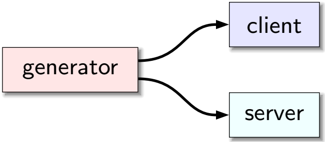
- Things get more interesting when we start moving **channels** around.



- The syntax '$\bar{x}$' represents the **server-end** of a mobile channel-type.
  - **Shared** mobiles are represented as '$x+$'
- The resulting expression here indicates a system in which a mobile 'b' is communicated **internally** over some mobile channel-bundle '$x, \bar{x}$', but which **never escapes**.

Higher Order Communication

- Things get more interesting when we start moving **channels** around.

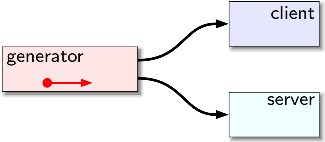



```
PROC generator (CHAN CT.FOO! out.c!,
                CHAN CT.FOO? out.s!)
  CT.FOO? svr:
  CT.FOO! cli:
  SEQ
    cli, svr := MOBILE CT.FOO
    PAR
      out.c ! cli
      out.s ! svr
:
```

- The syntax '$\bar{x}$' represents the **server-end** of a mobile channel-type.
  - **Shared** mobiles are represented as '$x+$'
- The resulting expression here indicates a system in which a mobile 'b' is communicated **internally** over some mobile channel-bundle '$x, \bar{x}$', but which **never escapes**.

Higher Order Communication

- Things get more interesting when we start moving **channels** around.

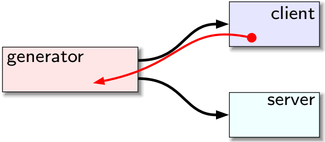



```
PROC generator (CHAN CT.FOO! out.c!,
                CHAN CT.FOO? out.s!)
  CT.FOO? svr:
  CT.FOO! cli:
  SEQ
    cli, svr := MOBILE CT.FOO
    PAR
      out.c ! cli
      out.s ! svr
:
```

- The syntax '$\bar{x}$' represents the **server-end** of a mobile channel-type.
  - **Shared** mobiles are represented as '$x+$'
- The resulting expression here indicates a system in which a mobile 'b' is communicated **internally** over some mobile channel-bundle '$x, \bar{x}$', but which **never escapes**.

Higher Order Communication

- Things get more interesting when we start moving **channels** around.

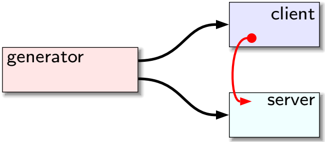



```
PROC generator (CHAN CT.FOO! out.c!,
                      CHAN CT.FOO? out.s!)
  CT.FOO? svr:
  CT.FOO! cli:
  SEQ
    cli, svr := MOBILE CT.FOO
    PAR
      out.c ! cli
      out.s ! svr
:
```

- The syntax '$\bar{x}$' represents the **server-end** of a mobile channel-type.
  - **Shared** mobiles are represented as '$x+$'
- The resulting expression here indicates a system in which a mobile 'b' is communicated **internally** over some mobile channel-bundle '$x, \bar{x}$', but which **never escapes**.

Higher Order Communication

- Things get more interesting when we start moving **channels** around.

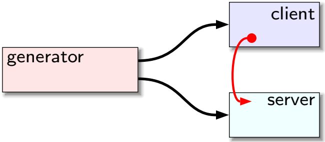



```
PROC generator (CHAN CT.FOO! out.c!,
                          CHAN CT.FOO? out.s!)
  CT.FOO? svr:
  CT.FOO! cli:
  SEQ
    cli, svr := MOBILE CT.FOO
    PAR
      out.c ! cli
      out.s ! svr
:
```

- The syntax '$\bar{x}$' represents the **server-end** of a mobile channel-type.
  - **Shared** mobiles are represented as '$x+$'
- The resulting expression here indicates a system in which a mobile 'b' is communicated **internally** over some mobile channel-bundle '$x, \bar{x}$', but which **never escapes**.

Higher Order Communication

- Things get more interesting when we start moving **channels** around.



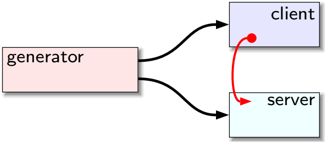
$$\textit{mobility}\ \mathrm{GEN} = \{\langle out.c!^{x}, out.s!^{\bar{x}}\rangle\}$$

```
PROC generator (CHAN CT.FOO! out.c!,
                CHAN CT.FOO? out.s!)
  CT.FOO? svr:
  CT.FOO! cli:
  SEQ
    cli, svr := MOBILE CT.FOO
    PAR
      out.c ! cli
      out.s ! svr
:
```

- The syntax '$\bar{x}$' represents the **server-end** of a mobile channel-type.
  - **Shared** mobiles are represented as '$x+$'
- The resulting expression here indicates a system in which a mobile 'b' is communicated **internally** over some mobile channel-bundle '$x, \bar{x}$', but which **never escapes**.

Higher Order Communication

- Things get more interesting when we start moving **channels** around.



$$\textit{mobility}\ \mathrm{GEN} = \{\langle out.c!^{x}, out.s!^{\bar{x}}\rangle\}$$

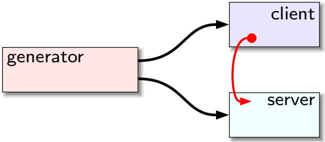
$$\textit{mobility}\ \mathrm{CLI} = \{\langle in?^{a}, a!^{b}\rangle\}$$

```
PROC client (CHAN CT.FOO! in?)
  CT.FOO! cli:
  MOBILE THING v:
  SEQ
    in ? cli
    ... initialise ‘v’
    cli[c] ! v
:
```

- The syntax ‘$\bar{x}$’ represents the **server-end** of a mobile channel-type.
  - **Shared** mobiles are represented as ‘$x+$’
- The resulting expression here indicates a system in which a mobile ‘b’ is communicated **internally** over some mobile channel-bundle ‘$x, \bar{x}$’, but which **never escapes**.

Higher Order Communication

- Things get more interesting when we start moving **channels** around.



$$mobility\ \mathrm{GEN} = \{\langle out.c!^{x}, out.s!^{\bar{x}}\rangle\}$$

$$mobility\ \mathrm{CLI} = \{\langle in?^{a}, a!^{b}\rangle\}$$

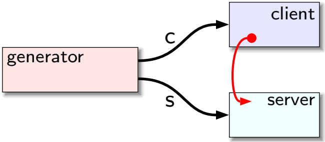
$$mobility\ \mathrm{SVR} = \{\langle in?^{\bar{d}}, \bar{d}?^{e}\rangle\}$$

```
PROC server (CHAN CT.FOO? in?)
  CT.FOO? svr:
  MOBILE THING x:
  SEQ
    in ? svr
    svr[c] ? x
    ... use 'x'
:
```

- The syntax '$\bar{x}$' represents the **server-end** of a mobile channel-type.
  - **Shared** mobiles are represented as '$x+$'
- The resulting expression here indicates a system in which a mobile 'b' is communicated **internally** over some mobile channel-bundle '$x, \bar{x}$', but which **never escapes**.

Higher Order Communication

- Things get more interesting when we start moving **channels** around.




```
CHAN CT.FOO! c:
CHAN CT.FOO? s:
PAR
  generator (c!, s!)
  client (c?)
  server (s?)
```

$$mobility\ \mathrm{GEN} = \{\langle out.c!^{x}, out.s!^{\bar{x}}\rangle\}$$

$$mobility\ \mathrm{CLI} = \{\langle in?^{a}, a!^{b}\rangle\}$$

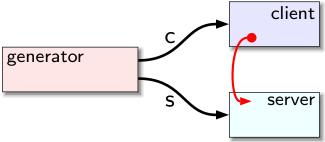
$$mobility\ \mathrm{SVR} = \{\langle in?^{\bar{d}}, \bar{d}?^{e}\rangle\}$$

$$mobility = \{\langle c!^{x}, s!^{\bar{x}}\rangle, \langle c?^{a}, a!^{b}\rangle, \langle s?^{\bar{d}}, \bar{d}?^{e}\rangle\} \setminus \{c, s\}$$

- The syntax '$\bar{x}$' represents the **server-end** of a mobile channel-type.
  - **Shared** mobiles are represented as '$x+$'
- The resulting expression here indicates a system in which a mobile 'b' is communicated **internally** over some mobile channel-bundle '$x, \bar{x}$', but which **never escapes**.

Higher Order Communication

- Things get more interesting when we start moving **channels** around.




```
CHAN CT.FOO! c:
CHAN CT.FOO? s:
PAR
  generator (c!, s!)
  client (c?)
  server (s?)
```

$$mobility\ \mathrm{GEN} = \{\langle out.c!^{x}, out.s!^{\bar{x}}\rangle\}$$

$$mobility\ \mathrm{CLI} = \{\langle in?^{a}, a!^{b}\rangle\}$$

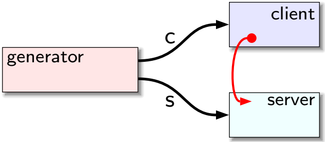
$$mobility\ \mathrm{SVR} = \{\langle in?^{\bar{d}}, \bar{d}?^{e}\rangle\}$$

$$\begin{aligned} mobility &= \{\langle c!^{x}, s!^{\bar{x}}\rangle, \langle c?^{a}, a!^{b}\rangle, \\ &\qquad \langle s?^{\bar{d}}, \bar{d}?^{e}\rangle\} \setminus \{c, s\} \\ &= \{\langle x!^{b}\rangle, \langle \bar{x}?^{d}\rangle\} \end{aligned}$$

- The syntax '$\bar{x}$' represents the **server-end** of a mobile channel-type.
  - **Shared** mobiles are represented as '$x+$'
- The resulting expression here indicates a system in which a mobile 'b' is communicated **internally** over some mobile channel-bundle '$x, \bar{x}$', but which **never escapes**.

Higher Order Communication

- Things get more interesting when we start moving **channels** around.




```
CHAN CT.FOO! c:
CHAN CT.FOO? s:
PAR
  generator (c!, s!)
  client (c?)
  server (s?)
```

$$\textit{mobility}\ \mathrm{GEN} = \{\langle \textit{out}.c!^{x}, \textit{out}.s!^{\bar{x}}\rangle\}$$

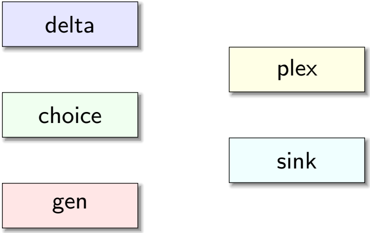
$$\textit{mobility}\ \mathrm{CLI} = \{\langle \textit{in}?^{a}, a!^{b}\rangle\}$$

$$\textit{mobility}\ \mathrm{SVR} = \{\langle \textit{in}?^{\bar{d}}, \bar{d}?^{e}\rangle\}$$

$$\begin{aligned} \textit{mobility} &= \{\langle c!^{x}, s!^{\bar{x}}\rangle, \langle c?^{a}, a!^{b}\rangle, \langle s?^{\bar{d}}, \bar{d}?^{e}\rangle\} \setminus \{c, s\} \\ &= \{\langle x!^{b}\rangle, \langle \bar{x}?^{d}\rangle\} \end{aligned}$$

- The syntax '$\bar{x}$' represents the **server-end** of a mobile channel-type.
  - **Shared** mobiles are represented as '$x+$'
- The resulting expression here indicates a system in which a mobile 'b' is communicated **internally** over some mobile channel-bundle '$x, \bar{x}$', but which **never escapes**.

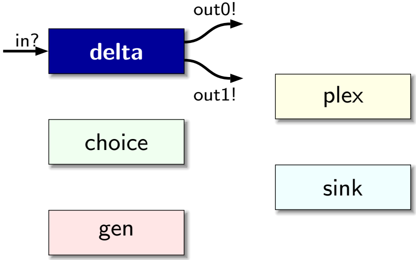
Using Mobile Analysis



- When **composed** in parallel, with **renaming** for parameter passing and avoiding capture, this gives the mobility set:

$$\mathit{mobility}\ \mathrm{net} = \{\langle A?^a, X!^a\rangle, \langle A?^b, p!^b\rangle, \langle B?^c, q!^c\rangle, \langle B?^d, r!^d\rangle$$
$$\langle s!^e\rangle, \langle p?^f, Y!^f\rangle, \langle q?^g, Y!^g\rangle, \langle r?^h\rangle, \langle s?^h\rangle\} \setminus \{p, q, r, s\}$$

Using Mobile Analysis

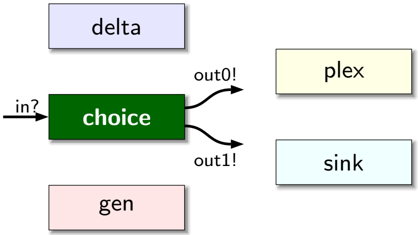


$$\textit{mobility}\ \mathrm{delta} = \{\langle in?^a, out0!^a\rangle,\ \langle in?^b, out1!^b\rangle\}$$

- When **composed** in parallel, with **renaming** for parameter passing and avoiding capture, this gives the mobility set:

$$\textit{mobility}\ \mathrm{net} = \{\langle A?^a, X!^a\rangle, \langle A?^b, p!^b\rangle, \langle B?^c, q!^c\rangle, \langle B?^d, r!^d\rangle \langle s!^e\rangle, \langle p?^f, Y!^f\rangle, \langle q?^g, Y!^g\rangle, \langle r?^h\rangle, \langle s?^h\rangle\} \setminus \{p, q, r, s\}$$

Using Mobile Analysis



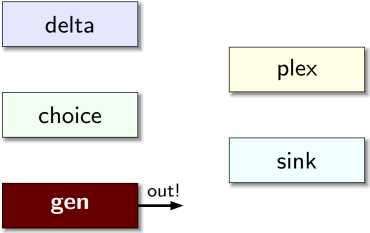
$$\textit{mobility}\ \mathrm{delta} = \{\langle \mathit{in}?^a, \mathit{out}0!^a\rangle, \langle \mathit{in}?^b, \mathit{out}1!^b\rangle\}$$

$$\textit{mobility}\ \mathrm{choice} = \{\langle \mathit{in}?^a, \mathit{out}0!^a\rangle, \langle \mathit{in}?^b, \mathit{out}1!^b\rangle\}$$

- When **composed** in parallel, with **renaming** for parameter passing and avoiding capture, this gives the mobility set:

$$\textit{mobility}\ \mathrm{net} = \{\langle A?^a, X!^a\rangle, \langle A?^b, p!^b\rangle, \langle B?^c, q!^c\rangle, \langle B?^d, r!^d\rangle \langle s!^e\rangle, \langle p?^f, Y!^f\rangle, \langle q?^g, Y!^g\rangle, \langle r?^h\rangle, \langle s?^h\rangle\} \setminus \{p, q, r, s\}$$

Using Mobile Analysis



$$\mathit{mobility}\ \mathrm{delta} = \{\langle \mathit{in}?^a, \mathit{out}0!^a\rangle, \langle \mathit{in}?^b, \mathit{out}1!^b\rangle\}$$

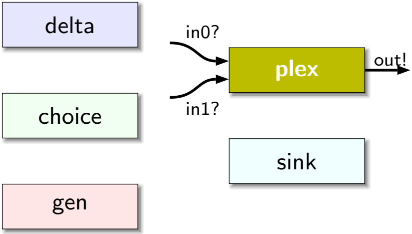
$$\mathit{mobility}\ \mathrm{choice} = \{\langle \mathit{in}?^a, \mathit{out}0!^a\rangle, \langle \mathit{in}?^b, \mathit{out}1!^b\rangle\}$$

$$\mathit{mobility}\ \mathrm{gen} = \{\langle \mathit{out}!^a\rangle\}$$

- When **composed** in parallel, with **renaming** for parameter passing and avoiding capture, this gives the mobility set:

$$\mathit{mobility}\ \mathrm{net} = \{\langle A?^a, X!^a\rangle, \langle A?^b, p!^b\rangle, \langle B?^c, q!^c\rangle, \langle B?^d, r!^d\rangle \langle s!^e\rangle, \langle p?^f, Y!^f\rangle, \langle q?^g, Y!^g\rangle, \langle r?^h\rangle, \langle s?^h\rangle\} \setminus \{p, q, r, s\}$$

Using Mobile Analysis



$$\mathit{mobility}\ \mathrm{delta} = \{\langle in?^a, out0!^a\rangle, \langle in?^b, out1!^b\rangle\}$$

$$\mathit{mobility}\ \mathrm{choice} = \{\langle in?^a, out0!^a\rangle, \langle in?^b, out1!^b\rangle\}$$

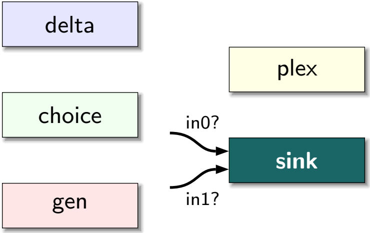
$$\mathit{mobility}\ \mathrm{gen} = \{\langle out!^a\rangle\}$$

$$\mathit{mobility}\ \mathrm{plex} = \{\langle in0?^a, out!^a\rangle, \langle in1?^b, out!^b\rangle\}$$

- When **composed** in parallel, with **renaming** for parameter passing and avoiding capture, this gives the mobility set:

$$\mathit{mobility}\ \mathrm{net} = \{\langle A?^a, X!^a\rangle, \langle A?^b, p!^b\rangle, \langle B?^c, q!^c\rangle, \langle B?^d, r!^d\rangle \langle s!^e\rangle, \langle p?^f, Y!^f\rangle, \langle q?^g, Y!^g\rangle, \langle r?^h\rangle, \langle s?^h\rangle\} \setminus \{p, q, r, s\}$$

Using Mobile Analysis



$$\textit{mobility}\ \mathrm{delta} = \{\langle \textit{in}?^a, \textit{out}0!^a\rangle, \langle \textit{in}?^b, \textit{out}1!^b\rangle\}$$

$$\textit{mobility}\ \mathrm{choice} = \{\langle \textit{in}?^a, \textit{out}0!^a\rangle, \langle \textit{in}?^b, \textit{out}1!^b\rangle\}$$

$$\textit{mobility}\ \mathrm{gen} = \{\langle \textit{out}!^a\rangle\}$$

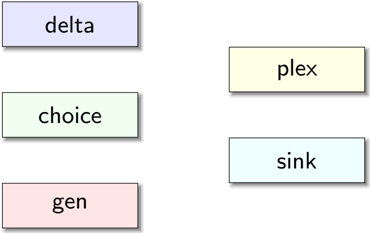
$$\textit{mobility}\ \mathrm{plex} = \{\langle \textit{in}0?^a, \textit{out}!^a\rangle, \langle \textit{in}1?^b, \textit{out}!^b\rangle\}$$

$$\textit{mobility}\ \mathrm{sink} = \{\langle \textit{in}0?^a\rangle, \langle \textit{in}1?^b\rangle\}$$

- When **composed** in parallel, with **renaming** for parameter passing and avoiding capture, this gives the mobility set:

$$\textit{mobility}\ \mathrm{net} = \{\langle A?^a, X!^a\rangle, \langle A?^b, p!^b\rangle, \langle B?^c, q!^c\rangle, \langle B?^d, r!^d\rangle \langle s!^e\rangle, \langle p?^f, Y!^f\rangle, \langle q?^g, Y!^g\rangle, \langle r?^h\rangle, \langle s?^h\rangle\} \setminus \{p, q, r, s\}$$

Using Mobile Analysis



$$\textit{mobility}\ \text{delta} = \{\langle \textit{in}?^a, \textit{out}0!^a\rangle, \langle \textit{in}?^b, \textit{out}1!^b\rangle\}$$

$$\textit{mobility}\ \text{choice} = \{\langle \textit{in}?^a, \textit{out}0!^a\rangle, \langle \textit{in}?^b, \textit{out}1!^b\rangle\}$$

$$\textit{mobility}\ \text{gen} = \{\langle \textit{out}!^a\rangle\}$$

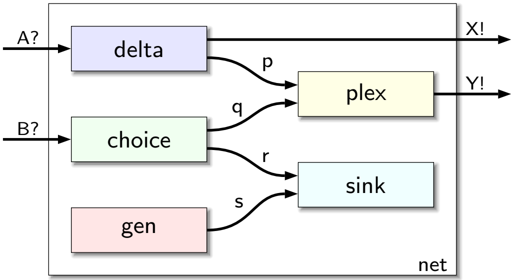
$$\textit{mobility}\ \text{plex} = \{\langle \textit{in}0?^a, \textit{out}!^a\rangle, \langle \textit{in}1?^b, \textit{out}!^b\rangle\}$$

$$\textit{mobility}\ \text{sink} = \{\langle \textit{in}0?^a\rangle, \langle \textit{in}1?^b\rangle\}$$

- When **composed** in parallel, with **renaming** for parameter passing and avoiding capture, this gives the mobility set:

$$\textit{mobility}\ \text{net} = \{\langle A?^a, X!^a\rangle, \langle A?^b, p!^b\rangle, \langle B?^c, q!^c\rangle, \langle B?^d, r!^d\rangle \langle s!^e\rangle, \langle p?^f, Y!^f\rangle, \langle q?^g, Y!^g\rangle, \langle r?^h\rangle, \langle s?^h\rangle\} \setminus \{p, q, r, s\}$$

Using Mobile Analysis



$$\textit{mobility}\ \mathrm{delta} = \{\langle \textit{in}?^a, \textit{out}0!^a\rangle, \langle \textit{in}?^b, \textit{out}1!^b\rangle\}$$

$$\textit{mobility}\ \mathrm{choice} = \{\langle \textit{in}?^a, \textit{out}0!^a\rangle, \langle \textit{in}?^b, \textit{out}1!^b\rangle\}$$

$$\textit{mobility}\ \mathrm{gen} = \{\langle \textit{out}!^a\rangle\}$$

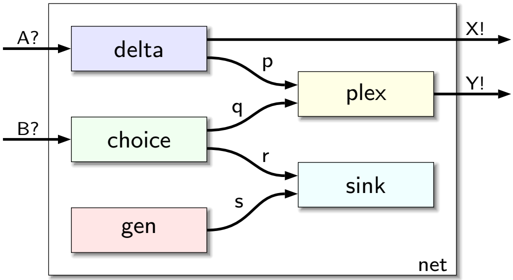
$$\textit{mobility}\ \mathrm{plex} = \{\langle \textit{in}0?^a, \textit{out}!^a\rangle, \langle \textit{in}1?^b, \textit{out}!^b\rangle\}$$

$$\textit{mobility}\ \mathrm{sink} = \{\langle \textit{in}0?^a\rangle, \langle \textit{in}1?^b\rangle\}$$

- When **composed** in parallel, with **renaming** for parameter passing and avoiding capture, this gives the mobility set:

$$\textit{mobility}\ \mathrm{net} = \{\langle A?^a, X!^a\rangle, \langle A?^b, p!^b\rangle, \langle B?^c, q!^c\rangle, \langle B?^d, r!^d\rangle \langle s!^e\rangle, \langle p?^f, Y!^f\rangle, \langle q?^g, Y!^g\rangle, \langle r?^h\rangle, \langle s?^h\rangle\} \setminus \{p, q, r, s\}$$

Using Mobile Analysis



$$\mathit{mobility}\ \mathrm{delta} = \{\langle \mathit{in}?^a, \mathit{out}0!^a\rangle, \langle \mathit{in}?^b, \mathit{out}1!^b\rangle\}$$

$$\mathit{mobility}\ \mathrm{choice} = \{\langle \mathit{in}?^a, \mathit{out}0!^a\rangle, \langle \mathit{in}?^b, \mathit{out}1!^b\rangle\}$$

$$\mathit{mobility}\ \mathrm{gen} = \{\langle \mathit{out}!^a\rangle\}$$

$$\mathit{mobility}\ \mathrm{plex} = \{\langle \mathit{in}0?^a, \mathit{out}!^a\rangle, \langle \mathit{in}1?^b, \mathit{out}!^b\rangle\}$$

$$\mathit{mobility}\ \mathrm{sink} = \{\langle \mathit{in}0?^a\rangle, \langle \mathit{in}1?^b\rangle\}$$

- When **composed** in parallel, with **renaming** for parameter passing and avoiding capture, this gives the mobility set:

$$\mathit{mobility}\ \mathrm{net} = \{\langle A?^a, X!^a\rangle, \langle A?^b, p!^b\rangle, \langle B?^c, q!^c\rangle, \langle B?^d, r!^d\rangle \\ \langle s!^e\rangle, \langle p?^f, Y!^f\rangle, \langle q?^g, Y!^g\rangle, \langle r?^h\rangle, \langle s?^h\rangle\} \setminus \{p, q, r, s\}$$

Using Mobile Analysis

- **Hiding** the internal channels gives:

$$\xrightarrow{\backslash\{p\}} \{\langle A?^a, X!^a\rangle, \langle A?^b, Y!^b\rangle, \langle B?^c, q!^c\rangle, \langle B?^d, r!^d\rangle, \langle s!^e\rangle, \langle q?^g, Y!^g\rangle, \langle r?^h\rangle, \langle s?^h\rangle\}$$

$$\xrightarrow{\backslash\{q\}} \{\langle A?^a, X!^a\rangle, \langle A?^b, Y!^b\rangle, \langle B?^c, Y!^c\rangle, \langle B?^d, r!^d\rangle, \langle s!^e\rangle, \langle r?^h\rangle, \langle s?^h\rangle\}$$

$$\xrightarrow{\backslash\{r\}} \{\langle A?^a, X!^a\rangle, \langle A?^b, Y!^b\rangle, \langle B?^c, Y!^c\rangle, \langle B?^d\rangle, \langle s!^e\rangle, \langle s?^h\rangle\}$$

$$\xrightarrow{\backslash\{s\}} \{\langle A?^a, X!^a\rangle, \langle A?^b, Y!^b\rangle, \langle B?^c, Y!^c\rangle, \langle B?^d\rangle\}$$

- Which indicates that mobiles arriving on **A** escape on **X** and **Y**; and that mobiles arriving on **B** escape on **Y** or are consumed internally.
  - by what is not present: no mobiles received on A are discarded internally; and that no internally generated mobiles escape.

Using Mobile Analysis

- **Hiding** the internal channels gives:

$$\xrightarrow{\backslash\{p\}} \quad \{\langle A?^a, X!^a\rangle, \langle A?^b, Y!^b\rangle, \langle B?^c, q!^c\rangle, \langle B?^d, r!^d\rangle, \langle s!^e\rangle, \langle q?^g, Y!^g\rangle, \langle r?^h\rangle, \langle s?^h\rangle\}$$

$$\xrightarrow{\backslash\{q\}} \quad \{\langle A?^a, X!^a\rangle, \langle A?^b, Y!^b\rangle, \langle B?^c, Y!^c\rangle, \langle B?^d, r!^d\rangle, \langle s!^e\rangle, \langle r?^h\rangle, \langle s?^h\rangle\}$$

$$\xrightarrow{\backslash\{r\}} \quad \{\langle A?^a, X!^a\rangle, \langle A?^b, Y!^b\rangle, \langle B?^c, Y!^c\rangle, \langle B?^d\rangle, \langle s!^e\rangle, \langle s?^h\rangle\}$$

$$\xrightarrow{\backslash\{s\}} \quad \{\langle A?^a, X!^a\rangle, \langle A?^b, Y!^b\rangle, \langle B?^c, Y!^c\rangle, \langle B?^d\rangle\}$$

- Which indicates that mobiles arriving on **A** escape on **X** and **Y**; and that mobiles arriving on **B** escape on **Y** or are consumed internally.
  - by what is not present: no mobiles received on A are discarded internally; and that no internally generated mobiles escape.

Using Mobile Analysis

- **Hiding** the internal channels gives:

$$\xrightarrow{\backslash\{p\}} \{\langle A?^a, X!^a\rangle, \langle A?^b, Y!^b\rangle, \langle B?^c, q!^c\rangle, \langle B?^d, r!^d\rangle, \langle s!^e\rangle, \langle q?^g, Y!^g\rangle, \langle r?^h\rangle, \langle s?^h\rangle\}$$

$$\xrightarrow{\backslash\{q\}} \{\langle A?^a, X!^a\rangle, \langle A?^b, Y!^b\rangle, \langle B?^c, Y!^c\rangle, \langle B?^d, r!^d\rangle, \langle s!^e\rangle, \langle r?^h\rangle, \langle s?^h\rangle\}$$

$$\xrightarrow{\backslash\{r\}} \{\langle A?^a, X!^a\rangle, \langle A?^b, Y!^b\rangle, \langle B?^c, Y!^c\rangle, \langle B?^d\rangle, \langle s!^e\rangle, \langle s?^h\rangle\}$$

$$\xrightarrow{\backslash\{s\}} \{\langle A?^a, X!^a\rangle, \langle A?^b, Y!^b\rangle, \langle B?^c, Y!^c\rangle, \langle B?^d\rangle\}$$

- Which indicates that mobiles arriving on **A** escape on **X** and **Y**; and that mobiles arriving on **B** escape on **Y** or are consumed internally.
  - by what is not present: no mobiles received on A are discarded internally; and that no internally generated mobiles escape.

Using Mobile Analysis

- **Hiding** the internal channels gives:

$$\xrightarrow{\backslash\{p\}} \{\langle A?^a, X!^a\rangle, \langle A?^b, Y!^b\rangle, \langle B?^c, q!^c\rangle, \langle B?^d, r!^d\rangle, \langle s!^e\rangle, \langle q?^g, Y!^g\rangle, \langle r?^h\rangle, \langle s?^h\rangle\}$$

$$\xrightarrow{\backslash\{q\}} \{\langle A?^a, X!^a\rangle, \langle A?^b, Y!^b\rangle, \langle B?^c, Y!^c\rangle, \langle B?^d, r!^d\rangle, \langle s!^e\rangle, \langle r?^h\rangle, \langle s?^h\rangle\}$$

$$\xrightarrow{\backslash\{r\}} \{\langle A?^a, X!^a\rangle, \langle A?^b, Y!^b\rangle, \langle B?^c, Y!^c\rangle, \langle B?^d\rangle, \langle s!^e\rangle, \langle s?^h\rangle\}$$

$$\xrightarrow{\backslash\{s\}} \{\langle A?^a, X!^a\rangle, \langle A?^b, Y!^b\rangle, \langle B?^c, Y!^c\rangle, \langle B?^d\rangle\}$$

- Which indicates that mobiles arriving on **A** escape on **X** and **Y**; and that mobiles arriving on **B** escape on **Y** or are consumed internally.
  - by what is **not present**: no mobiles received on **A** are discarded internally; and that no internally generated mobiles escape.

Using Mobile Analysis

- **Hiding** the internal channels gives:

$$\xrightarrow{\backslash\{p\}} \quad \{\langle A?^a, X!^a\rangle, \langle A?^b, Y!^b\rangle, \langle B?^c, q!^c\rangle, \langle B?^d, r!^d\rangle, \langle s!^e\rangle, \langle q?^g, Y!^g\rangle, \langle r?^h\rangle, \langle s?^h\rangle\}$$

$$\xrightarrow{\backslash\{q\}} \quad \{\langle A?^a, X!^a\rangle, \langle A?^b, Y!^b\rangle, \langle B?^c, Y!^c\rangle, \langle B?^d, r!^d\rangle, \langle s!^e\rangle, \langle r?^h\rangle, \langle s?^h\rangle\}$$

$$\xrightarrow{\backslash\{r\}} \quad \{\langle A?^a, X!^a\rangle, \langle A?^b, Y!^b\rangle, \langle B?^c, Y!^c\rangle, \langle B?^d\rangle, \langle s!^e\rangle, \langle s?^h\rangle\}$$

$$\xrightarrow{\backslash\{s\}} \quad \{\langle A?^a, X!^a\rangle, \langle A?^b, Y!^b\rangle, \langle B?^c, Y!^c\rangle, \langle B?^d\rangle\}$$

- Which indicates that mobiles arriving on **A** escape on **X** and **Y**; and that mobiles arriving on **B** escape on **Y** or are consumed internally.
  - by what is **not present**: no mobiles received on **A** are discarded internally; and that no internally generated mobiles escape.

Using Mobile Analysis

- **Hiding** the internal channels gives:

$$\xrightarrow{\setminus\{p\}} \{\langle A?^a, X!^a\rangle, \langle A?^b, Y!^b\rangle, \langle B?^c, q!^c\rangle, \langle B?^d, r!^d\rangle, \langle s!^e\rangle, \langle q?^g, Y!^g\rangle, \langle r?^h\rangle, \langle s?^h\rangle\}$$

$$\xrightarrow{\setminus\{q\}} \{\langle A?^a, X!^a\rangle, \langle A?^b, Y!^b\rangle, \langle B?^c, Y!^c\rangle, \langle B?^d, r!^d\rangle, \langle s!^e\rangle, \langle r?^h\rangle, \langle s?^h\rangle\}$$

$$\xrightarrow{\setminus\{r\}} \{\langle A?^a, X!^a\rangle, \langle A?^b, Y!^b\rangle, \langle B?^c, Y!^c\rangle, \langle B?^d\rangle, \langle s!^e\rangle, \langle s?^h\rangle\}$$

$$\xrightarrow{\setminus\{s\}} \{\langle A?^a, X!^a\rangle, \langle A?^b, Y!^b\rangle, \langle B?^c, Y!^c\rangle, \langle B?^d\rangle\}$$

- Which indicates that mobiles arriving on **A** escape on **X** and **Y**; and that mobiles arriving on **B** escape on **Y** or are consumed internally.
  - by what is **not present**: no mobiles received on **A** are discarded internally; and that no internally generated mobiles escape.

Conclusions

- A semantic model that can be used to reason about the **escape** of **mobile** items within an occam-π system.
  - handles the **movement** of **mobile channels** and subsequent communication on then.
  - can be used to reason about **safety properties** of systems involving mobiles – and to inform optimisation or distribution.
- The paper contains more details on the **semantics**, as well as more **complex examples** taken from real systems.
- **Future work** includes:
  - generation of these models **automatically** by the compiler.
  - **tools** to manipulate models (not as complex as FDR).
  - complete **denotational** semantics.
  - application of the techniques to other process-oriented systems.

Conclusions

- A semantic model that can be used to reason about the **escape** of **mobile** items within an occam-π system.
  - handles the **movement** of **mobile channels** and subsequent communication on then.
  - can be used to reason about **safety properties** of systems involving mobiles – and to inform optimisation or distribution.
- The paper contains more details on the **semantics**, as well as more **complex examples** taken from real systems.
- **Future work** includes:
  - generation of these models **automatically** by the compiler.
  - **tools** to manipulate models (not as complex as FDR).
  - complete **denotational** semantics.
  - application of the techniques to other process-oriented systems.

Conclusions

- A semantic model that can be used to reason about the **escape** of **mobile** items within an occam-π system.
  - handles the **movement** of **mobile channels** and subsequent communication on then.
  - can be used to reason about **safety properties** of systems involving mobiles – and to inform optimisation or distribution.
- The paper contains more details on the **semantics**, as well as more **complex examples** taken from real systems.
- **Future work** includes:
  - generation of these models **automatically** by the compiler.
  - **tools** to manipulate models (not as complex as FDR).
  - complete **denotational** semantics.
  - application of the techniques to other process-oriented systems.

The End

- **Any questions**?

- This work was funded by EPSRC grant EP/D061822/1.
- Thanks to the anonymous reviewers and colleagues for their feedback.

RMoX

PLAS
Programming Languages and Systems

EPSRC Engineering and Physical Sciences Research Council

University of Kent | Computing

References

C.A.R. Hoare.
Communicating Sequential Processes.
Prentice-Hall, London, 1985.
ISBN: 0-13-153271-5.

Frederick R. M. Barnes and Carl G. Ritson.
Checking process-oriented operating system behaviour using CSP and refinement.
In *PLOS 2009*. ACM.
To Appear.

R. Milner.
Communicating and Mobile Systems: the Pi-Calculus.
Cambridge University Press, 1999.
ISBN: 0-52165-869-1.

Mobility Refinement

- With the ordinary semantic models, we have a notion of **refinement**.
- no reason why one should not exist for the **mobility** model presented here:

$$P \sqsubseteq_M Q \quad \equiv \quad \mathit{mobility}\ Q \subseteq \mathit{mobility}\ P$$

- The informal interpretation is that **Q** is "less leaky" than **P**, when it comes to mobile escape.
  - some fudge required in the subset operation: e.g. $\{\langle c?^x \rangle\}$ refines $\{\langle c?^x, d!^x \rangle\}$, as does $\{\langle d!^y \rangle\}$.
  - can arise in an implementation that *copies* data between mobiles.

Expansive Hiding

- Hiding is not always an **reducing** operation:
  - can easily **blow-up**, reflecting the different possibilities for mobiles.

$$\begin{aligned} & \{\langle A?^a, c!^a\rangle, \langle B?^b, c!^b\rangle, \langle c?^f, X!^f\rangle, \langle c?^g, Y!^g\rangle, \langle c?^h, Z!^h\rangle\} \\ \xrightarrow{\backslash\{c\}} \quad & \{\langle A?^a, X!^a\rangle, \langle A?^a, Y!^a\rangle, \langle A?^a, Z!^a\rangle, \\ & \quad \langle B?^b, X!^b\rangle, \langle B?^b, Y!^b\rangle, \langle B?^b, Z!^b\rangle\} \end{aligned}$$

- Worse-case is limited by type compatibility.

Denotational Semantics

- Alphabets (for any particular **occam-π** process):
  - **output** channels: $\Sigma^{!}$, **input** channels: $\Sigma^{?}$, such that $\Sigma = \Sigma^{!} \cup \Sigma^{?}$.
  - also grouped by **type**: Σ_t, where t is a valid occam-π **protocol** and $t \in \mathbb{T}$, where $\mathbb{T}$ is the set of valid occam-π protocols.
    - following on: $\Sigma_t = \Sigma^{!}_t \cup \Sigma^{?}_t$, and $\forall\, t : \mathbb{T} \cdot \Sigma_t \subseteq \Sigma$.
  - for **shared** mobiles: $\Sigma_{+} = \Sigma^{!}_{+} \cup \Sigma^{?}_{+}$.
- Primitive processes:

$$\begin{aligned}
\mathit{mobility}\ \mathit{SKIP} &= \langle\rangle \\
\mathit{mobility}\ \mathit{STOP} &= \langle\rangle \\
\mathit{mobility}\ \mathrm{div} &= \mathit{mobility}\ \mathit{CHAOS} = \\
&\qquad \{\langle C!^{a}\rangle \mid C \in \Sigma^{!}\} \cup \{\langle D?^{x}\rangle \mid D \in \Sigma^{?}\} \cup \\
&\qquad\quad \{\langle C?^{v}, D!^{v}\rangle \mid \forall\, t : \mathbb{T} \cdot (C, D) \in \Sigma^{?}_t \times \Sigma^{!}_t)\}
\end{aligned}$$

Denotational Semantics

- Choice:

$$mobility\ (P \mathrel{\Box} Q) = (mobility\ P) \cup (mobility\ Q)$$
$$mobility\ (P \sqcap Q) = (mobility\ P) \cup (mobility\ Q)$$

- Interleaving and parallelism:

$$mobility\ (P \parallel Q) = (mobility P) \cup (mobility\ Q)$$

- Hiding:

$$\begin{aligned}
mobility\ (P \setminus x) = \{ & M \frown N[\alpha/\beta] \mid \\
& (M \frown \langle x!^{\alpha}\rangle, \langle x?^{\beta}\rangle \frown N) \in mobility\ P \times mobility P\} \cup \\
& ((mobility\ P) - (\{F \frown \langle x!^{\alpha}\rangle \mid F \frown \langle x!^{\alpha}\rangle \in mobility\ P\} \\
& \quad \cup \{\langle x?^{\beta}\rangle \frown G \mid \langle x?^{\beta}\rangle \frown G \in mobility\ P\})) \cup \\
& \{H \mid (H \frown \langle x!^{\alpha}\rangle) \in mobility\ P \wedge (\langle x?^{\beta}\rangle \frown I) \notin mobility\ P \wedge H \neq \langle\rangle\} \cup \\
& \{J \mid (\langle x?^{\beta}\rangle \frown J) \in mobility\ P \wedge (J \frown \langle x!^{\alpha}\rangle) \notin mobility\ P \wedge J \neq \langle\rangle\}
\end{aligned}$$